Dipartimento di Informatica
Università degli Studi di Verona

3 marzo 2013

Dispense del corso di Sistemi Embedded di Rete

Ambra Fausti (autrice delle dispense)
Davide Quaglia (docente del corso)

Indice

Capitolo 1

Introduzione

Questa dispensa si propone di fornire un approccio scientifico alla progettazione, realizzazione e verifica di applicazioni basate su sistemi embedded di rete come quelle per domotica, automazione industriale, sanitá, automotive, controllo e gestione delle risorse ambientali.
Verranno descritti nel Capitolo 2 i principali protocolli di comunicazione per sistemi embedded di rete, focalizzando l'attenzione sulle reti wireless e le reti di campo.
Nel Capitolo 3 verranno affrontate le metodologie per la progettazione, la simulazione e la verifica di networked embedded system.
Infine, nel Capitolo 4, verrá presentato il problema della progettazione di sistemi di controllo mediante rete e verranno proposte alcune tecniche di progettazione. La trattazione avverrá attingendo dalla letteratura scientifica e da attivitá di ricerca direttamente svolte dall'Universitá di Verona.

Capitolo 2

Protocolli di comunicazione per sistemi embedded di rete

2.1 Reti di sensori wireless

2.1.1 Definizione e applicazione delle reti di sensori

Per rete di sensori (Wireless Sennor Network, WSN) si indica una determinata tipologia di rete che, caratterizzata da una architettura distribuita, è realizzata da un insieme di dispositivi elettronici autonomi (alimentati a batteria) in grado di prelevare dati dall'ambiente circostante e di comunicare tra loro. Questi piccoli dispositivi sono prodotti e distribuiti in massa, hanno un costo di produzione trascurabile e sono caratterizzati da dimensioni e pesi molto ridotti. Ogni sensore ha una riserva di energia limitata e non rinnovabile e, una volta messo in opera, deve lavorare autonomamente: per questo motivo tali dispositivi devono mantenere costantemente i consumi molto bassi, al fine di avere un maggior ciclo di vita. Per ottenere la maggior quantità possibile di dati occorre effettuare una massiccia distribuzione di sensori in modo da avere una alta densità (fino a 20 nodi/m^3) e far sí che i nodi siano tutti vicini tra loro, condizione necessaria affinchè possano comunicare. I nodi sensore all'interno di una rete hanno la possibilità di collaborare tra loro dal momento che sono provvisti di un processor-on-board; grazie a quest'ultimo ciascun nodo, invece di inviare dati grezzi ai nodi responsabili della raccolta dei dati, puó effettuare delle semplici elaborazioni e trasmettere solo i dati richiesti e già elaborati. La comunicazione, realizzata tramite tecnologia wireless a corto raggio, è solitamente di tipo asimmetrico in quanto i nodi comuni inviano le informazioni raccolte ad uno o piú nodi speciali della rete, detti nodi sink, i quali hanno lo scopo di raccogliere i dati e trasmetterli tipicamente ad un server o ad un calcolatore. Una comunicazione puó avvenire autonomamente da parte del nodo quando si verifica un dato evento, o puó venire indotta dal nodo sink tramite l'invio di una query

verso i nodi interessati.
L'obbiettivo di una rete di sensori è quello di istanziare da 100 a 1000 nodi, che forniscono un'alta risoluzione, distanziati di pochi centimetri fino a un massimo di 10 metri. Sono caratterizzati da una comunicazione di tipo wireless poichè risulta piú facile da usare in quanto non bisogna usare cavi. Si ricorda che in questi sistemi la principale forma di consumo di potenza è la trasmissione (per trasmettere un bit servono 1000 istruzioni!).
Lo Smartdust (polvere intelligente) è un ipotetico network costituito da microscopici sistemi elettromeccanici (MEMS) messi in comunicazione mediante un sistema wireless capaci di rilevare, per esempio, luce, temperatura oppure vibrazioni. Le unità costituenti il network sono ipotizzate della dimensione di un granello di sabbia: da questo ne deriva il nome. Dentro ci sono uno o piú sensori o attuatori. In genere l'attuatore è un po' piú grande perchè è necessario un circuito di potenza. Sono molto piccoli quelli che sfruttano la luce o la temperatura, mentre per l'umidità serve qualcosa di piú ricercato.
I Micro Electro Mechanical Systems (MEMS), sono un insieme di dispositivi di varia natura (meccanici, elettrici ed elettronici) integrati in forma altamente miniaturizzata su uno stesso substrato di silicio, che coniugano le proprietà elettriche degli integrati a semiconduttore con proprietà optomeccaniche. Ne è un esempio l'accelerometro, che è un sistema massa-molla che fornisce l'accelerazione e puó essere costruito direttamente sul silicio. Nello stesso substrato di silicio c'è anche il processore, la memoria e un bus.
Gli Imote2, in figura Figura 2.1, sono stati sviluppati dai laboratori: sono nodi ad alte performance sviluppati per la ricerca in ambito di Wireless Sensor Network. Sono i dispositivi piú performanti tra quelli sviluppati nel campo delle WSN e risulta netta la differenza con altri tipi di motes. Il

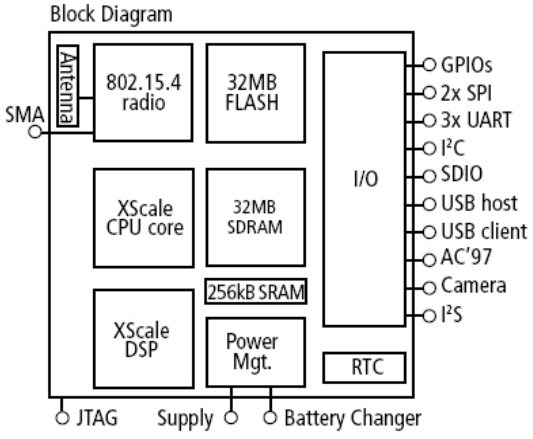


Figura 2.1: Imote2

dispositivo imote2 presenta una struttura modulare essendo composto di di-

verse boards interconnesse attraverso connettori posti su entrambi i layers delle schede.
Sono forniti di:

Processor board: include il microprocessore PXA271 (da 13 a 416 MHz) che integra 256KB di SRAM, 32 MB di FLASH e 32 MB di SDRAM, ed il modulo radio (802.15.4 , 2.4 GHz radioband). Il processore integra alcui set di I/O rendendo la piattaforma estremamente dinamica supportando diversi standard di comunicazioni (I2C, SPI, UART);

Sensor board: questa scheda integra vari sensori tra cui un sensore di luce, un sensore di temperatura e umidità , un ulteriore sensore di temperatura, un accelerometro ed infine un convertitore analogico/digitale;

Debug board: consente di installare il software nella memoria flash del microcontrollore e consente il debug attraverso l'interfaccia JTAG;

Battery board: permette di alimentare il mote qualora non sia connesso tramite USB al PC. I sensori grandi e potenti sono difficili da installare sul campo, consumano molto, sono costosi e risultano infine inefficienti. Inoltre il loro utilizzo puó essere indesiderabile e pericoloso.

2.1.2 Tipologie di reti wireless

Una classificazione delle reti wireless puó essere fatta in base all'area coperta dal segnale trasmesso dai dispositivi. Le reti wireless piú piccole vengono chiamate Wireless Body Area Network (WBAN). Una Body Area Network serve per connettere dispositivi "indossabili" sul corpo umano come ad esempio possono essere i componenti di un computer (auricolari); idealmente una rete di questo tipo supporta l'auto-configurazione, facendo cosí in modo che l'inserimento e la rimozione di un dispositivo dalla BAN risultino essere trasparenti all'utente. Aumentando il raggio d'azione di una rete si passa alle Wireless Personal Area Network (WPAN). Il raggio di comunicazione delle PAN si aggira intorno ai 10 metri e l'obiettivo di queste reti è quello di consentire a dispositivi vicini di condividere dinamicamente informazioni. Pertanto, mentre una BAN si dedica all'interconnessione dei dispositivi indossabili da una persona, una PAN è operativa nella immediate vicinanze degli utenti. In una Personal Area Network è possibile perció connettere dispositivi portatili tra loro oppure con stazioni fisse, ad esempio per accedere ad Internet, sincronizzare o scambiari dati e contenuti multimediali su computer portatili, desktop e palmari. Lo standard piú utilizzato per la realizzazione delle reti Body/Personal Area Network è il Bluetooth. Un raggio d'azione maggiore hanno invece le Wireless Local Area Network (WLAN), che consentono la comunicazione tra dispositivi distanti tra loro, anche alcune centinaia di metri, e risultando cosí adeguate per realizzare soluzioni di interconnessione in ambienti chiusi o tra edifici adiacenti. La famiglia degli

standard piú diffusi per la realizzazione di questo genere di interconnessioni è IEEE 802.11x e puó essere considerata l'alternativa wireless alle tradizionali LAN basate su IEEE 802.3/Ethernet. Le Wireless Metropolitan Area Network (WMAN) hanno invece un raggio d'azione di circa 10 Km, fornendo cosí un accesso a banda larga ad aree residenziali anche piuttosto estese. La tecnologia piú accreditata per la realizzazione di reti Wireless MAN è WiMAX, basata sullo standard IEEE 802.16. Quello sopra proposto non è peró l'unico modo possibile per classificare le reti wireless. Ad esempio è possibile confrontare le tecnologie in base al consumo e alla velocità di trasmissione (data-rate). In Figura 2.2 vengono messi in relazione il range che un protocollo wireless riesce a coprire e la velocitá di trasmissione. In generale, alti data-rate implicano consumi elevati e viceversa.

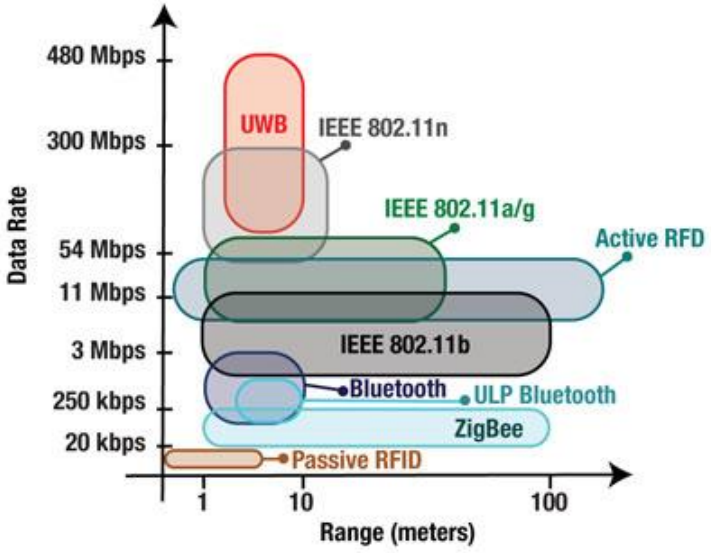


Figura 2.2: Rapporto tra velocità e range coperto dei protocolli wireless

2.1.3 Applicazioni di WSN

Le applicazioni piú comuni delle reti di sensori,alcune delle quali sono mostrate in Figura 2.3 sono:

- Monitoraggio ambientale (Enviromental and habitat monitoring)
  In campo ambientale le applicazioni possono essere molteplici: è possibile monitorare movimenti di animali oppure studiare particolari habitat, come il mare, il terreno o l'aria impiegando, ad esempio, sistemi di monitoraggio di agenti inquinanti. Appartengono a questo settore anche lo studio di eventi naturali catastrofici quali incendi, tornado, terremoti ed eruzioni vulcaniche.

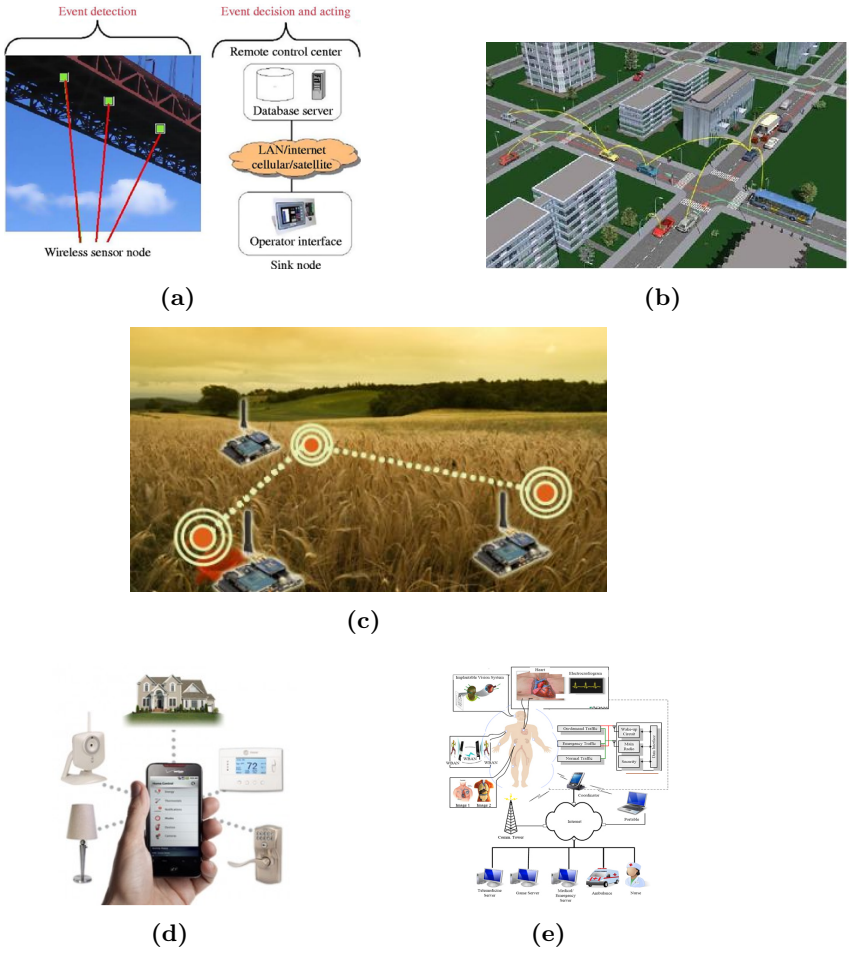


Figura 2.3: Applicazioni di reti di sensori

- Monitoraggio di strutture (Structural Health Monitoring)
  Le reti di sensori posizionate sulle strutture rilevano lo stato di salute di edifici, ponti, case sottoposte a sollecitazioni esterne; in alternativa, potrebbero essere utilizzate anche per misurare difetti strutturali di componenti.

- Controllo del traffico veicolare (Traffic control)
  Un sistema di sensori finalizzato al monitoraggio del traffico controlla il passaggio di automobili, analizza la velocità e l'intensità del traffico ed individua eventuali blocchi o situazioni anomale.

- Sorveglianza di edifici (Infrastructure control)
  Questo tipo di reti puó essere utilizzato come ausilio per la sorveglianza di centri commerciali o di luoghi a rischio, come stazioni ferroviarie o aereoporti.

- Sorveglianza militare (Militar control)
  Le reti di sensori sono state utilizzate in principio per questo scopo. Le loro applicazioni spaziano dal monitoraggio sullo stato o la posizione delle forze sul campo di battaglia alla sorveglianza di luoghi strategici. Possono inoltre essere dispiegate in luoghi ostili al fine di raccogliere informazioni sugli spostamenti del nemico.

- Monitoraggio di apparecchiature industriali (Industrial sensing)
  I sensori wireless possono essere applicati a macchinari industriali per analizzarne il comportamento dei componenti sottoposti a stress meccanico, migliorarne le performance o prevenire rotture e guasti.

- Monitoraggio agricolo (Agriculture sensing)
  Nel settore agricolo le reti WSN vengono utilizzate per il monitoraggio di particolari situazioni ambientali; in particolar modo la cosiddetta "agricoltura di precisione" utilizza le reti di sensori per rilevare in tempo reale il livello di pesticidi nell'acqua o nei terreni agricoli.

- Applicazioni personali (Personal sensoring)
  Nel settore della domotica le reti di sensori riescono a fornire servizi all'utente all'interno della propria abitazione, ad esempio informandolo tempestivamente di eventuali guasti. I sensori possono essere introdotti negli apparecchi elettrici e questi, interagendo tra loro e con una rete esterna o Internet stesso, permettono all'utente di comandare facilmente gli elettrodomestici a distanza.

- Applicazioni mediche (Personal Health Care)
  Le reti wireless sono oggi impiegate anche per monitorare pazienti, eseguire valutazioni diagnostiche, amministrare farmaci in ospedale e monitorare a distanza dati fisiologici quali la frequenza cardiaca o la pressione sanguigna.

2.1.4 Sfide di progettazione

Le principali sfide di progettazione sono le seguenti:

Efficienza energetica: Una delle sfide principali delle reti di sensori è l'efficienza energetica, poiché, affinchè queste reti siano accettate a livello applicativo, occorre che la loro batteria duri molto tempo, ovvero che l'utente non si debba preoccupare di cambiare la batteria continuamente. La maggior componente di consumo è data dalla trasmissione: l'obbiettivo è quindi trovare protocolli di trasmissione piu efficiente. Inoltre bisogna gestire la raccolta di energia dall'ambiente, ad esempio la luce, come succede per i pannelli fotovoltaici, le vibrazioni, i flussi di calore l'induzione elettromagnetica;

Prontezza: I sensori non devono essere attivi tutto il tempo: non è necessario. Bisogna attivarli solo nel momento in cui occorrono. Ad esempio per un sensore che manda una temperatura ogni minuto è sufficiente che stia sveglio 10 millisecondi, il tempo di percepire la temperatura, elaborarla e inviarla, dopodichè puó essere addormentato fino al minuto successivo. Se viene messo a dormire quindi si introduce il problema di responsiveness, ovvero il tempo che impiegherebbe il sistema per comunicare col nodo, dovendo aspettare che questo si svegli. In applicazioni di tipo di real time, nelle quali è necessaria una risposta in corrispondenza di un evento, bisogna trovare un compromesso tra risparmio e prontezza.

Robustezza: I sensori vengono messi in ambienti diversi e non sempre sicuri: per questo motivo è tollerata una percentuale di guasto. L'obbietivo è quello che, nel ciclo di vita di un sistema, anche se alcuni sensori si rompono, ci sia una forma di instradamento dei messaggi che permetta di ottenere comunque le informazioni necessarie. Infatti per queste applicazioni non è importante comunicare con un nodo in particolare ma ottenere l'informazione da tutto il sistema. Un esempio sono le reti peer to peer: si chiede un'informazione indipendentemente da chi me la possa fornire.

Sinergia: La capacità di collaborare ovvero la sinergia tra i vari nodi è fondamentale. Si vorrebbe idealmente che la capacità computazionale totale sia superiore alla somma delle capacità dei singoli nodi, ottenendo un guadagno dall'interazione tra questi. In sostanza il fatto che si osservi un fenomeno da piú punti di vista fornisce informazioni piú precise.

Scalabilità: Un esempio di un sistema non scalabile è dato dai primi telefoni: questi venivano collegati con un filo a due a due, ovvero per mettere in comunicazione un telefono con un altro si usava un filo diretto tra i due, quindi per far comunicare n telefoni occorrevano $n * (n - 1)$ fili. L'aumento dei fili era quadratico rispetto all'aumento dei telefoni che volevano comunicare tra di loro. Questo è un esempio di sistema non scalabile. Un sistema scalabile invece sfrutta la comunicazione, ovvero tutti i nodi sono collegati ad un centro stella e la comunicazione viene gestita da quest'ultimo, al fine di avere un aumento lineare di collegamenti rispetto al numero di nodi.
Se ad esempio non è stato implementato un protocollo di routing efficiente, si rischia, nel caso in cui ci siano tanti nodi, che sia piú l'energia spesa dai nodi che per organizzarsi che l'energia spesa per mandare i

dati che servono all'utente.

Eterogeneità: Una rete di sensori generalmente non è composta di nodi dello stesso tipo, anche perchè non esiste un' architettura ottima per qualsiasi applicazione. L'eterogeneità consiste nel far parlare tra di loro nodi diversi, con processori differenti e capacità di memoria differenti.

Configurazione automatica: La configurazione automatica di un sensore è una caratteristica obbligatoria per reti composte da molti nodi. Ciascun nodo dev'essere in grado di configurare la propria topologia di rete e saper svolgere le funzioni di localizzazione, sincronizzazione, e calibrazione. Devono inoltre saper coordinare le comunicazioni tra di loro. Tutto questo ha un impatto positivo sia sui costi dell'installazione e sia sulla robustezza.

Adattamento: Le WSN non possono essere ottimizzate a priori. L'ambiente non è prevedibile e puó cambiare drasticamente. I protocolli per reti di sensori dovrebbero essere adattati e saper adeguarsi online.

Design: La progettazione di una WSN deve scontrarsi con due alternative: essere molto specifici, ovvero sfruttare le caratteristiche specifiche applicative per migliorare le performance oppure essere piú generali per rendere piú facile l'adattamento in diverse situazioni a spese delle performance. Si cerca quindi un trade-off tra le due alternative. Sono richieste inoltre metodologie per il riuso, la modularità e l'adattamento run-time.

Sicurezza e privacy: è necessario il supporto di criteri di sicurezza per applicazioni critiche, per evitare ad esempio il sabotaggio per sistemi di monitoraggio strutturale. Bisogna inoltre garantire la privacy, quando ad esempio si gestiscono dati medici.

2.1.5 Il livello datalink

Il datalink è il livello 2 del modello di riferimento ISO/OSI. Gli obbiettivi del livello datalink sono la comunicazione affidabile ed efficiente tra due luoghi a e b con un canale in mezzo fisico (e non un altro nodo). Il livello fisico ha definito il concetto di bit e il concetto che i bit, se ricevuti, sono ricevuti in maniera ordinata, ovvero con lo stesso ordine col quale sono mandati. Il livello datalink sotto questa ipotesi ha l'obbiettivo di costruire una comunicazione efficiente, ovvero col minor numero di risorse, ed efficace, ovvero che funzioni, sul canale fisico. Il framing nasce per rendere la comunicazione efficace: il concetto di pacchetto nasce in questo livello, ovvero raggruppare

i bit in PDU (Protocol Data Unit) di livello 2. La definizione di frame è quindi PDU di livello 2. A livello 2 viene fatto anche error checking, ovvero rilevazione degli errori. Alcuni protocolli prevedono anche che a livello 2 vengano inviati ack per trasmissione ricevuta oppure ritrasmissione in mancanza di questi. Ethernet ad esempio non prevede questa opzione, invece tutti i protocolli wireless lo prevedono, perchè la probabilit1'a di errore è alta. Il livello datalink gestisce inoltre la politica di accesso al canale se piú di due stazioni utilizzano lo stesso mezzo fisico. Questo prevede due problematiche:

- indirizzamento, se in tanti nodi accedono allo stesso canale, per permettere la comunicazione tra due nodi devo assegnare degli indirizzi ai nodi;

- arbitraggio del canale, non si puo parlare contemporaneamente sul canale perchè si sovrapporrebbero i bit.

Al livello superiore puó venire fornito un servizio non confermato, ovvero senza acknowledge e non connesso, cioè non c'è una connessione tra due nodi quando questi si parlano: ogni volta che viene mandata una PDU di livello due al livello superiore bisogna scrivere un indirizzo e inviarlo quindi come fosse una lettera. Un esempio è il caso del 802.3, ovvero ethernet). In alternativa puó essere un servizio confermato ma non connesso: ogni volta che viene mandato una PDU deve ricevere l'acknowledge, e se non la riceve ritrasmette la PDU fino ad un certo numero di volte, ma ogni PDU è indipendente dalle altre e quindi ogni PDU deve avere davanti l'intestazione del destinatario. Questo succede in diversi casi wireless come il wifi oppure uno standard per reti di sensori che è l'802.15.4 Molto piú raro invece il caso di un servizio confermato e connesso ad esempio per l' 802.16. Un esempio di servizio orientato alle connessioni fuori dal mondo delle reti è la telefonata: una volta che si inizia una telefonata si crea una connessione punto punto e quindi non è piu necessario mettere un'intestazione davanti al pacchetto perchè il destinatario è già stato definito nel momento in cui inizia la telefonata. Questo servizio peró costa troppo, quindi è piú conveniente fornire la connessione a livello piú alto, nel livello trasporto.
Lo scopo del framing è quindi quello di migliorare l'utilizzo del canale in caso di piú di due nodi che lo condividono. In presenza di un canale punto punto, è possibile mandare i bit uno alla volta; con un canale condiviso, davanti all'informazione, bisogna mettere un'intestazione davanti al bit trasmesso. L'intestazione è un numero almeno di 8 bit (per l'802.15.4 sono 48 bit) e andrebbe messa davanti ad un solo bit: si deduce che l'intestazione costa piú dell'informazione. Ecco allora che viene introdotto il concetto di pacchetto, per una ragione quindi di costo e di ottimizzazione: mandando piú bit assieme è possibile liberare piú velocemente l'utilizzo del canale. Un altro motivo che ha spinto a raggruppare i bit assieme è la verifica di errori.

Per fare error detection, ovvero per sapere se i bit ricevuti sono veri o sono errati, è necessario appendere ai bit di dato, dei bit di verifica. Sarebbe illogico pensare di aggiungere 10 bit di verifica ad un solo bit. Rimane quindi la questione su come definire un frame. Infatti nel canale fisico non passano i bit bensí i segnali fisici. Il bit 0 e il bit 1 sono stati definiti, ma non è stato definito il "niente". Si immagini la fibraottica: 1 corrisponde a luce e 0 a spento. Quando non trasmette è spento, quindi potrebbe corrispondere ad uno zero. Analogamente se si trasmette un frame composto da un dato il quale ultimo byte è composto solo da zeri, si potrebbe pensare invece che non stia piú trasmettendo.
E' necessario definire quindi l'inizio e la fine del frame. In sostanza, quando un nodo vuole mandare dei dati, li passa a livello 2, gli mette davanti un header, necessario anche per l'indirizzamento, e mette anche una coda. Da notare che il livello 2 è l'unico caso in cui viene messa una coda alla fine del pacchetto.
Per realizzare questo sono stati proposti diversi modi:

- usare dei simboli fisici che non vengono usati nei dati. Ad esempio sul segnale elettrico spesso si usa il fronte di salita per l'1, il fronte di discesa per lo 0 e quindi posso definire tutto in alto o tutto in basso per la fine del frame;

- si forza uno spazio minimo tra due pacchetti consecutivi, ovvero appena finisce un pacchetto è noto che per n secondi non ne inizierà un altro, anche se in quel lasso di tempo sono comunque presenti dei bit;

- si definisce il bit stuffing, ovvero una sequenza di bit di dato particolare messi come inizio e fine, ma occorre anche trovare un sistema per fare in modo che una sequenza di dati nel pacchetto non venga confusa con questo flag. Viene fornito un esempio nella Figura 2.4

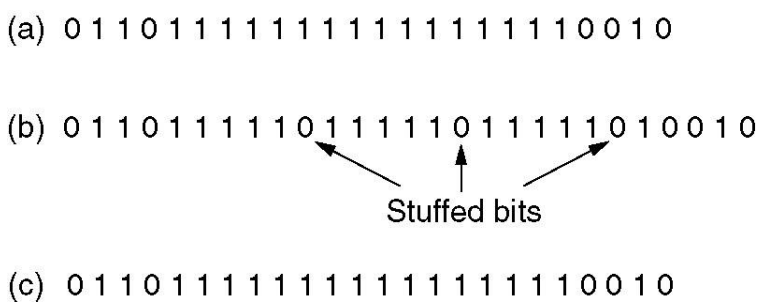


Figura 2.4: Bit stuffing

2.1.6 Error detection

In ricezione i bit possono essere capiti in maniera sbagliata; questo problema puó essere dovuto a questioni fisiche, come un livello di segnale troppo basso, oppure ad interferenze con altri nodi come il problema dell'hidden node, l'eco ecc. Gli errori non sono isolati, ma sono raggruppati in raffiche o burst, perchè le cause descritte sopra hanno una certa durata e quindi non coinvolgono solo un bit, ma ne coinvoglono diversi: questo rappresenta un vantaggio perchè ne facilita l'individuazione.
La detection degli errori si basa sul concetto di *distanza di Hamming*, definita come il numero di bit differenti tra due stringhe di pari lunghezza. La detection degli errori ha il compito di raggruppare i bit in stringhe, aggiungendo serie di bit che non facciano confondere una stringa errata con una stringa dell'alfabeto. Si hanno m bit in una stringa, si aggiungono r bit ridondanti per l'error detection : il totale, $n = m+r$, è il numero di bit che attraversno il canale.
Si definisce *code rate* la quantità m/n.
Rimane da definire con quale criterio aggiungere gli r bit.

Parity bit: Al trasmettitore si appende un solo bit, quindi il code rate è $m/(m+1)$.
Si appende un 1 se il numero di bit a 1 nel messaggio originale è pari, e quindi il numero di 1 diventa dispari.
si appende uno 0 se il numero di bit a 1 nel messaggio originale è dispari, e quindi il numero di 1 rimane dispari.
Se il ricevitore riceve un numero pari di 1, vuol dire che almeno un bit è stato cambiato e quindi il messaggio ricevuto è sicuramente sbagliato.
Se il ricevitore riceve un numero dispari di 1, non è possibile definire con certezza, con questo metodo, se il messaggio è corretto o meno.
Questo metodo, vista la poca affidabilità, non si usa mai nelle reti, si usa piú spesso nelle memorie perchè lí avvengono errori su un singolo bit.

Checksum: Scompone il messaggio originale in byte, ovvero in parole da 8 bit, e somma tutti byte. Il totale ricavato che occupa r bit è appeso al messaggio.
Il ricevitore fa il medesimmo calcolo e controlla se il risultato è uguale a quello che gli è stato inviato: in caso negativo vuol dire che c'è stato qualche errore.
Anche questo metodo non è molto affidabile perchè possono capitare errori tali da decrementare di una certa quantità il valore di un byte e aumentare della medesima quantità un altro byte, cosí da far rimanere immutata la somma totale.

Circular Redundancy Check (CRC): Si basa sulla divisione di polinomi: il messagio di m bit è visto come un vettore di coefficienti di un

polinomio di grado *m-1*. Si definisce questo polinomio *M(x)*. Si definisce poi il polinomio *R(x)* il resto ottenuto da $\left(\frac{x^r M(x)}{G(x)}\right)$ dove *G(x)* è il polinomio generatore e r è il numero di bit che si vuole appendere al messaggio originale di m bit. Si invia $x^r M(x) - R(x)$, quantitá divisibile esattamente per $G(x)$, al ricevitore. Questo divide il valore ottenuto per $G(x)$ e se il risultato è intero allora il valore ricevuto è corretto. Ha un basso costo di implementazione ed è molto piu potente di checksum, anche se non è perfetto, e per questo motivo è il piú usato.

2.1.7 Accesso al canale

Il canale puó essere punto punto o condiviso. Per i canali punto punto, che collegano quindi solo due nodi, non sussistono problemi di indirizzamentoo di accesso al canale perchè o parlano uno alla volta, se ad esempio si tratta di un canale radio, oppure il canale riesce a sostenere la comunicazione in entrambi i versi (full duplex).
Piú complicato è invece il canale condiviso perchè nono tanti i nodi che insistono sullo stesso canale.
La Figura 2.5 schematizza i principali metodi di accesso al canale.

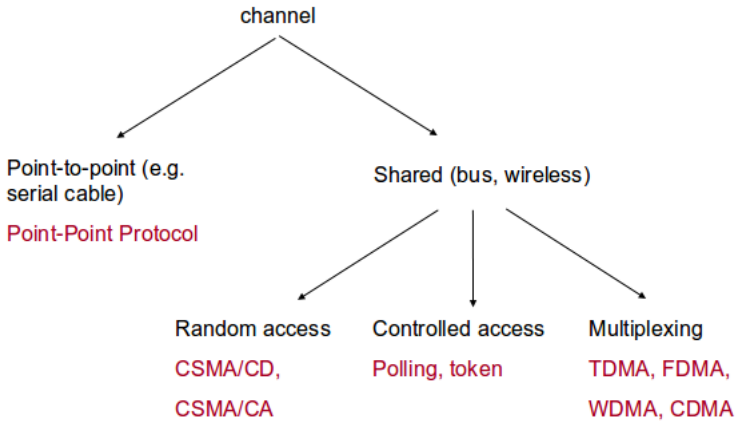


Figura 2.5: Accesso al canale

Ci sono diversi metodi per l'accesso al canale condiviso:

- Random access: è un approccio a tentativi e ad attese casuali ovvero non è definito in maniera deterministica quando un nodo parlerà. Le politiche di questo tipo che vanno per la maggiore sono il *CSMA*/CA

e il *CSMA*/CD. è statisticamente provato che questo metodo funziona nella maggior parte dei casi.

- Controlled access: C'è una struttura centrale che dice chi deve parlare e in quale momento. In questo contesto ci sono due principali approcci. Il primo utilizza dei protocolli Master/Slave che prevedono che ci sia un Master che ha l'autorità di scegliere quando un nodo puó parlare. Chiede al nodo se ha qualcosa da dire: se il nodo risponde si, puó parlare. Gli altri nodi non sono stati interrogati, e, anche se avessero qualcosa da dire, devono stare zitti. Quindi è il Master a garantire che i nodi parlino uno alla volta.
  Il secondo approccio tratta di un controllo distribuito attraverso il concetto di token. Si puó pensare al token come una palla: la palla è una sola e chi ha la palla in mano è l'unico a poter parlare. Una volta finito, il nodo puó passare la palla ad un altro nodo.

- Multiplexing: Il multiplexing divide il canale fisico in tanti sottocanali logici virtuali.
  Il canale è realizzato mediante la trasmissione di un segnale e i segnali hanno una certa frequenza: ragionando quindi in termini di frequenza, si puó trasmettere a una frequenza tale che le frequenze vicine non ne sentano l'influenza (concetto di radio FM). Idealmente è come se il mittente avesse tutto il canale radio per sé. è sufficiente quindi suddividere un certo range di frequenza in tanti range piú piccoli e far sí che ogni coppia di stazioni, che vuole parlare, utilizzi un specifico e piccolo canale in frequenza. Questa cosa puó essere fatta nel mondo radio, chiamata FDM (Frequency Division Multiplexing) mentre quando si usano le fibre ottiche è chiamata WDM (Wavelength Division Multiplexing); in quest'ultimo caso non si suddividono le frequenze, ma si trasmettono i bit nella fibra ottica con colori differenti.
  Un ulteriore approccio è il Time Division Multiplexing che divide il canale a turni temporali fissi (usato nei telefoni fissi attuali e i cellulari GSM): quindi ogni nodo per un lasso di tempo definito ha il canale tutto per sè .
  Per aumentare la capacità di un canale sono stati combinati l'aspetto in frequenza e l'aspetto nel tempo: sono metodologie complicate che vengono chiamate Code Division Multiplexing che sfruttano sia la suddivisione nel tempo che la suddivisione nella frequenza e queste vengono utilizzate per esempio nella terza generazione, dagli UMTS e successivi.
  Oggi, ad esempio per il WIFI 802.11n utilizzano un approccio frequenza + tempo + spazio.

Andando piú nel dettaglio delle trasmissioni wireless si ha un problema di interferenza, infatti ci sono molti dispositivi perchè si utilizzano delle frequenze

non licenziate, cioè non regolamentate. Il 2.4GHz, essendo una frequenza libera, dove trasmettono Wifi, bluetooth, rete di sensori ecc., è molto rumoroso. è da considerarsi anche il problema di perdita dovuta alla distanza (path loss), ovvero l'energia decresce con la distanza e decresce inoltre se nel tragitto incontra degli ostacoli. Le onde radio all'aumentare della frequenza assomigliano sempre di piú a delle onde luminose, questo è il motivo per cui una radio FM viene sentita anche al chiuso: utilizza delle frequenze piú basse che attraversano anche le pareti, e quindi lo spessore del muro non è critico per la lunghezza d'onda del campo di frequenza. Il wifi invece non si sente proprio perchè utilizza 2.4GHz che sono già delle microonde, molto piú vicino alle onde luminose che hanno una propagazione in linea retta e possono essere bloccate da muri anche non troppo spessi, perchè hanno una lunghezza d'onda molto piccola. In piu cè anche l'effetto delle riflessioni: le riflessioni di un segnale fanno sí che la stessa copia del segnale arrivi al ricevitore sfasato nel tempo.
Questi due aspetti generano il Bit Error Rate, una probabilità di errore sul bit non trascurabile, indipendentemente dalla tecnologia.
La probabilità di ricevere un bit corretto è *1-p*.
La probabilità di ricevere una PDU di lunghezza N è $(1-p)^n$,ovvero n istanze dello stesso fenomeno e perchè il frame sia corretto bisogna riceverle tutte.
Per questo motivo Ethernet non ha acknowledge, siccome la probabilità è prossima all'1, sarebbe inutile e ridondante. Mentre per il wifi si ha una probabilità di errore molto piú alta e quindi è opportuno usare un approccio con acknowledge. Inoltre questo valore è stato calcolato considerando tutte le problematiche elencate finora, ma oltre a queste, ce ne sono anche altre e quindi questo valore è destinato a scendere ulteriormente. Le ulteriori problematiche che possono presentarsi sono fondamentalmente due: il caso dell'hidden node e il caso dell'expose node. In Ethernet, il nodo sente il canale, se è libero e ha qualcosa da dire, parla, altrimenti aspetta.
Nel caso wireless questo non funziona, ovvero, risolve solo una parte del problema. Potrebbe verificarsi il fenomeno descritto nel caso (a) della Figura 2.6 :
Il nodo A vuole mandare un frame a B ma nello stesso momento C sta trasmettendo un frame ad un altro nodo e il suo raggio copre anche B.
A non sente che C sta parlando e quindi trasmette a B.
In B peró quel pacchetto andrà a collidere con la trasmissione di C.
Avremo dunque due danni: l'invalidazione del pacchetto da A a B e l'invalidazione del pacchetto da C a D.
Questa situazione viene chiamata fenomeno dell'hidden node.
Il caso (b), l'expose node, non produce un errore di trasmissione ma abbassa il rate di trasmissione:
A trasmette a B.
B vuole trasmettere a C e sente il canale, ma il canale non è libero perchè

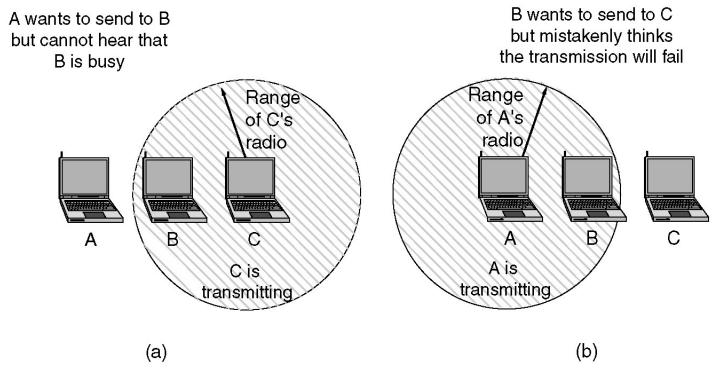


Figura 2.6: Hidden Node e Expose Node

sente che A sta trasmettendo e quindi aspetta.
La trasmissione da B a C non creerebbe collisioni con il pacchetto che A sta trasmettendo a B.
Questo fenomeno quindi abbassa il throughput, quantità di bit che una stazione riesce a mandare al secondo.
Dunque in questo contesto il *CSMA/CD* usato dall'Ethernet fallisce.
è opportuno usare il *CSMA/CA* e lo stop and wait ack. [1] Lo stop and wait ack vuole che il nodo mandi il pacchetto e poi si fermi, ovvero non puó mandare altri pacchetti finchè non ha ricevuto l'ack. Questa è la forma piú costosa e meno efficiente di acknowledge perchè spreca molto nell'attesa dell'arrivo di un frame, ma è il piú robusto. Opzionalmente si utilizza un'ulteriore meccanismo che si chiama *RTS /CTS* (Request to Send/Clear to Send).
Si riporta un esempio per spiegarne il funzionamento.
Ci sono 5 stazioni : A B C D E.
A trasmette a B
C vuole trasmettere a B e aspetta che il canale sia libero.
D fa parte del raggio di C e non di B
E fa parte del raggio di B e non di C .
C, una volta libero il canale, aspetta un tempo casuale prima di trasmettere. Scaduto questo tempo, invece di mandare direttamente il pacchetto, ne manda uno piú piccolo che serve a chiedere a B se ha il permesso di mandare un frame. Questo serve per evitare il fenomeno dell'Hidden node. Inoltre quando C manda il request specifica anche quanto sarà lungo il suo pay-

[1] La differenza tra *CA* e *CD* è la seguente: il *CD* appena sente libero inizia subito a parlare e ascolta il canale per vedere se quello che sente è effettivamente quello che ha trasmesso altrimenti vuol dire che cè stata una collisione. In tal caso è obbligato a fermarsi e aspettare un tempo casuale prima di ricominciare a trasmettere).
Il *CA* invece ascolta il canale e appena sente libero aspetta un tempo casuale per iniziare a trasmettere. Quando poi inizia a trasmettere non si mette in ascolto come il *CD*. Questo abbassa un pó il throughput, ma migliora la probabilità di successo.

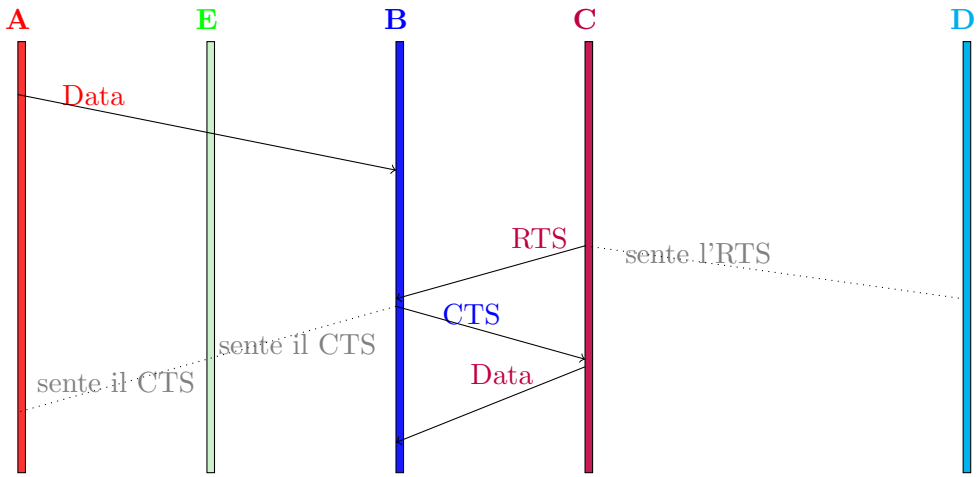


Figura 2.7: RTS/CTS

load. Questo serve affinchè D, che fa parte del raggio di C, senta che sta per essere trasmesso un pacchetto di lunghezza n e, quando B risponderà a Ccon un pacchetto *CTS*, che quindi autorizza l'invio, anche il nodo E ricaverà questa informazione. In questo modo D ed E anche senza ascoltare il canale sapranno che ci sarà una trasmissione di esattamente n byte.

2.2 Protocolli MAC per reti di sensori

Le reti broadcast hanno una particolarità per quanto riguarda il livello DataLink: occorre controllare l'accesso al canale condiviso.
In ogni rete broadcast il problema principale è quello di individuare chi deve usare il canale quando c'è un conflitto per utilizzarlo. Ci sono molti protocolli per risolvere questo problema.
I protocolli utilizzati per determinare a chi spetta la precedenza su un canale multiaccesso appartengono ad un sottolivello del livello DataLink, chiamato *MAC* (Medium Access control).

2.2.1 Sfide generali

Le sfide generali per il Medium Access control sono le seguenti:

Fairness. L'arbitraggio del canale deve essere "fair", non deve privilegiare nessuno. Questo non vuol dire che non ci siano priorità, ma dati due nodi di pari priorità il protocollo deve garantire che vengano trattati allo stesso modo.

Latenza. In certi meccanismi che hanno una natura real time, che devono rispettare determinate deadline, il protocollo MAC deve rispettare

queste necessità e di conseguenza deve considerare anche i problemi di latenza massima garantita.

Throughput. Si cerca di avere in ogni istante il canale occupato, cercando quindi di schedulare i pacchetti in maniera ottima, ottenendo un aumento del throughput; questo non è semplice perchè non si conoscono a priori le necessità future.

Nello specifico delle reti di sensori wireless ci sono delle problematiche in piú:

Efficienza di potenza. Le reti wireless sono piccoli nodi che vengono alimentati da una batteria oppure che utilizzano energie catturate dall'ambiente come vibrazioni, movimenti meccanici ecc. è quindi richiesto un basso consumo di energia. Per questa problematica, se è raro che il fenomeno hidden node si verifichi, l'RTS /CTS rappresenterebbe una perdita di tempo e di energia. In presenza di hidden node invece salva la situazione da un danno peggiore, ovvero ritrasmettere un pacchetto che ha subíto una collisione.

Scalabilità Una rete di sensori ha la forza di dover fare del lavoro che viene distribuito su un gran numero di sensori. Se ci sono tanti nodi la probabilità che due nodi vogliano trasmettere nello stesso istante è molto alta, e il sistema dev'essere in grado di gestirlo.

Vediamo inoltre quali sono le sorgenti di consumo di energia:

1. Collisioni, seguite dalla ritrasmissione del pacchetto;
2. Ascolto del canale, per vedere se è libero;
3. Overhearing, ascoltare pacchetti indirizzati da altri nodi;
4. Packetoverhead, avere un header grosso è uno spreco;
5. Pacchetti di controllo, ad esempio pacchetti RTS /CTS richiedono ulteriore energia, per essere mandati e ascoltati.

In generale l'arbitraggio di un canale, in qualunque modo venga fatto, ha un costo non trascurabile. L'unico tipo di arbitraggio che non ha un costo è quello a priori, cioè quando si conosce già la circolazione dei pacchetti futura.

2.2.2 Strategie proposte

Per risolvere i problemi descritti nella sezione precedente sono state avanzate delle strategie:

1. Wake up Radio (risveglio asincrono)
   Una seconda radio ascolta se sul canale c'è qualcosa; in caso affermativo manda un interrupt al canale principale. Questa radio consuma pochissimo perchè anche questa si addormenta e si risveglia solo quando rileva una presenza sul canale per mandare un segnale al sistema. In questo modo un nodo si accende solo quando arriva il segnale.

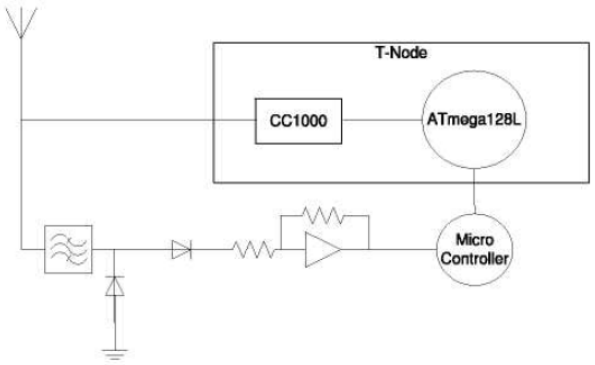


Figura 2.8: Wake up Radio

2. Polling asimmetrico.
   Ha un approccio Master/Slave. Viene implementato nell'802.15.4 .

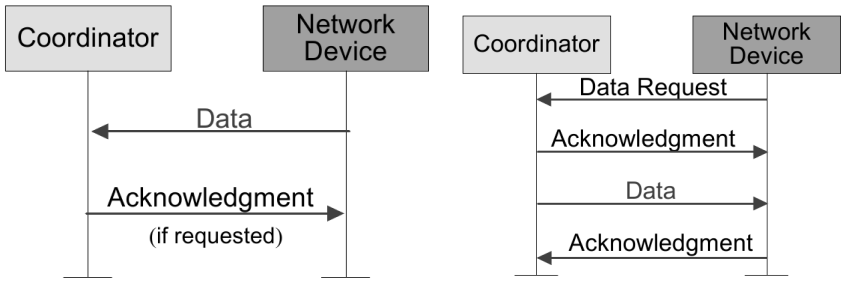


Figura 2.9: Polling asimmetrico

C'è un oggetto, chiamato coordinatore, a cui tutti i dispositivi low-power si agganciano. Il coordinatore è attaccato a una fonte di energia stabile ed è sempre sveglio. Gli altri nodi sono invece oggetti piccoli, economici, che consumano poco. Il coordinatore non puó mandare dati in qualsiasi momento perchè il network device potrebbe non essere sveglio. Succede invece che il network device appena si sveglia manda una richiesta al coordinatore per vedere se ci sono dati per lui, come descritto in Figura 2.9. Questo manda un acknowledge che contiene un bit posto a 0 se non sono presenti, permettendo cosí al nodo di rimettersi a dormire, oppure a 1 se invece sono presenti, seguito poi

dall'invio dei dati da parte del coordinatore.
è un approccio simile ad una casella di posta.

3. Basata su Timer.
   Un nodo si addormenta per un certo periodo fissato, e si risveglia solo quando questo intervallo è terminato. (risveglio periodico)
   Ci sono tanti approcci di questo tipo:

   - S-MAC: tutti i nodi si riaccendono periodicamente assieme. Man mano che aumenta questo periodo di sleep, aumenta la latenza. C'è una fase iniziale di sincronizzazione durante la quale tutti

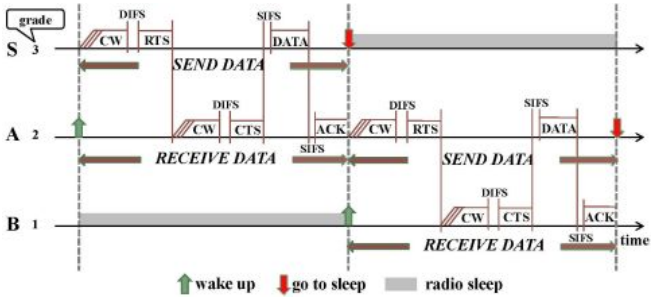


Figura 2.10: Approccio S-MAC

     i nodi fanno carrier sense, cioè ascoltano per un tempo casuale (perchè si parla di *CSMA/CA*) , e c'è un nodo che manda un pacchetto di sincronizzazione perchè i clock potrebbero essere sfasati. Dopodichè c'è la parte di *CSMA/CA* con *RTS/CTS*.
     Un ulteriore ottimizzazione prevede che quando qualcuno sente il *CTS* si addormenta, perchè tanto sa che qualcun altro ha ottenuto il canale.
     Il consumo maggiore quindi riguarda il periodo di listening, quando tutti devono stare accesi fino a quando qualcuno non inizia a parlare, l'overhead, durante il momento di sincronizzazione, e infine il consumo dovuto all'invio di *RTS/CTS*.

   - B-MAC,X-MAC,SCP-MAC: sono approcci basati su un risveglio casuale dove il risveglio dei singoli è indipendente da quello degli altri.
     Il Berkeley Mac Introduce un nuovo modo per sentire se il canale è libero che si chiama Clear Channel Assessment (CCA). Generalmente quando si vuole ascoltare se un canale è occupato non basta solo ascoltare se ci sono onde radio, ma bisogna prendere quest'onda radio, demodularla e ottenere i bit. Il B-MAC non ascolta quali bit sono presenti ma solo se in quella banda c'è traffico oppure no. Misura quindi l'SNR (Signal to noise Radio) che

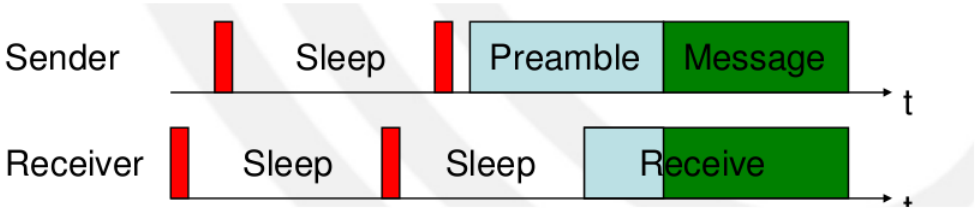


Figura 2.11: B-MAC

corrisponde al rapporto tra segnale e rumore, S/N: questo valore indica traffico se supera una certa soglia. La base di questo protocollo è che il ricevitore rimane sveglio per un tempo breve e periodico durante il quale effettua il CCA, in maniera del tutto asincrona dagli altri. Il periodo di addormentamento è uguale per tutti e viene dato dal protocollo; quello che non garantisce è che si risvegliano contemporaneamente, ovvero possono non essere allineati. Il periodo ha notevole importanza perchè l'invio dei dati funziona nel modo seguente: quando un nodo si risveglia e vuole mandare dei dati, non li manda subito; questi sono preceduti da un preambolo. Il preambolo dev'essere sufficientemente lungo affinchè tutti gli altri nodi si sveglino durante questo e si accorgano che il canale è occupato. Quindi il preambolo dev'essere piú lungo del periodo di sleep. Da questo consegue che i periodi di sleep devono essere uguali per tutti, anche se non allineati. Se il periodo di sleep è piú corto del preambolo vuol dire che in qualsiasi posizione temporale si trovi il preambolo, sicuramente capiterà a cavallo di due periodi di sleep. Quindi ciascun nodo si sveglia dopo il periodo di sleep, fa un CCA, si accorge del preambolo e rimane sveglio ad ascoltare il preambolo anche se il dato seguente non è destinato a lui (il preambolo non specifica chi è il ricevitore). Quando inizia il messaggio, se è rivolto a lui rimane sveglio e ascolta, altrimenti si rimette a dormire. In questo protocollo non viene gestito il problema dell'hidden terminal e neanche il meccanismo di trasmissione di piú pacchetti in sequenza.
L'X-Mac prevede che chi vuole mandare dati genera dei piccoli preamboli in cui è specificata la destinazione. Quindi un nodo si sveglia dal suo periodo di sleep, ascolta il canale, trova il preambolo e controlla se è il destinatario: se non lo è si rimette a dormire, altrimenti manda un ack al mittente che indica che si è accorto del preambolo e rimane sveglio per ascoltare il messaggio. Il mittente appena riceve l'ack interrompe il preambolo. quindi il preambolo è mediamente molto piú corto del preambolo del B-MAC. Nonostante nel B-MAC all'interno del preambolo non

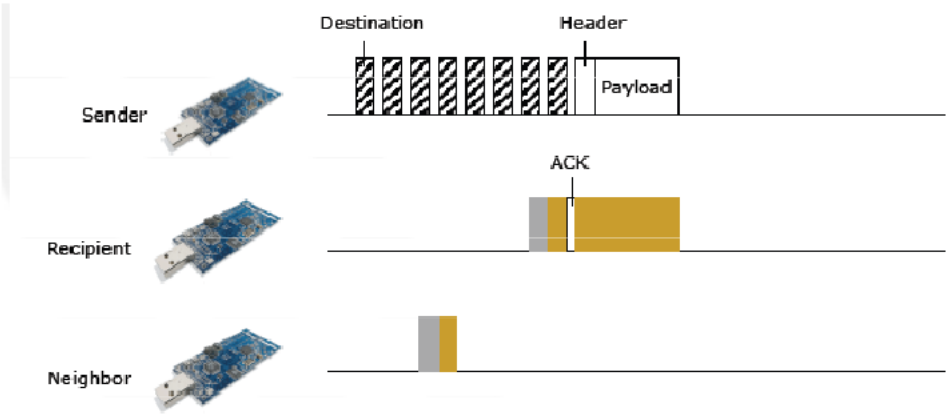


Figura 2.12: X-MAC

ci sia l'indirizzo di destinazione, i nodi eseguano un CCA, il che consuma molto di meno, e non ci sia l'ack da parte del destinatario, mediamente l'X-MAC consuma meno del B-MAC perchè riesce a addormentare i nodi piú frequentemente.
Un'ulteriore miglioria è introdotta dal protocollo SCP-MAC (Scheduled Channel polling). Per evitare lunghi preamboli, i nodi sono

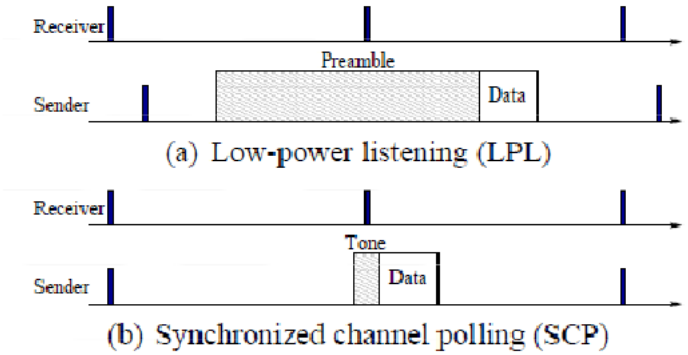


Figura 2.13: SCP-MAC

sincronizzati per cui tutti devono svegliarsi nello stesso istante e quindi il mittente puó mandare un preambolo piú corto. Bisogna vedere quanto costa sincronizzarli.

- TDMA (time division multiple access) : i nodi si risvegliano ognuno in uno slot di tempo assegnatogli, parlano col coordinatore e poi si riaddormentano. Non ci sono mai sovrapposizioni di slot. Il TDMA ha bisogno di un master: questo introduce un punto

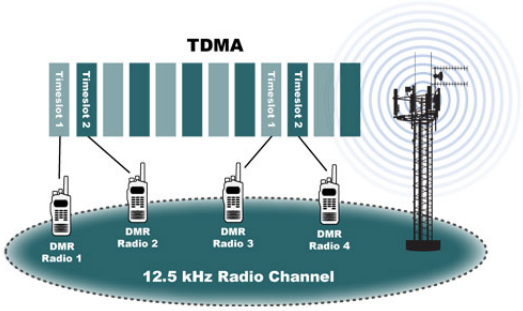


Figura 2.14: Approccio TDMA

di debolezza della rete perchè si dipende da un nodo, pericoloso nel caso di guasti ecc. Questo master manda dei segnali che dividono il tempo in cicli. Ogni ciclo di trasmissione inizia con una fase di sincronizzazione ed è seguito da intervalli assegnati ai vari nodi, durante i quali sono autorizzati a parlare. Ogni nodo quindi puó parlare solo in una certa fascia di tempo. Negli intervalli non assegnati i nodi possono addormentarsi. I vantaggi sono indubbiamente la maggiore densità dei dati, che porta ad avere un throughput massimo e pochissimi tempi morti, l'assenza del problema dell'hidden node, l'assenza di RTS/CTS e di preamboli perchè non necessari. Gli svantaggi sono dati dal vincolo della struttura, cioè dalla necessità di un master (se questo viene a mancare tutto il sistema crolla) e anche dalla fase di sincronizzazione iniziale, della quale non si puó fare a meno.

- Z-MAC: mette insieme due MAC: CSMA e TDMA. Si comporta come un CSMA quando il traffico è basso e fa uno switch in TDMA quando è alto.
  Il CSMA infatti è semplice, scala bene perchè non ha il master ma ha il problema dell'hidden node e l'overhead dovuto all' RTS-/CTS, che peró compaiono sopratutto nei momenti di alto traffico. Ecco che allora si ricorre al TDMA che evita naturalmente le collisioni ma per contro è complicato e richiede inoltre una sincronizzazione.

2.3 Standard 802.15.4

2.3.1 Introduzione

Nell'ambito delle comunicazioni wireless, e piú precisamente nelle wireless a corto raggio, l'IEEE ha definito una standard chiamato 802.15.4 . Questo standard è pensato per comunicazioni wireless di basso costo, bassa velocità e basso consumo energetico. è adatto a reti W-PAN a basso bit rate costituite da dispositivi alimentati tramite batterie che non possono essere sostituite frequentemente, come, ad esempio, i sensori. Le sue principali caratteristiche sono la capacità di coprire un raggio di azione di poche decine di metri offrendo fino ad un data-rate massimo di 250 Kbps. Offre una bassa potenza in trasmissione implicando un basso consumo energetico. Permette ai livelli superiori la presenza di livelli di rete che possono effettuare routing. Offre altresí la possibilità di poter utilizzare ACK e quindi di poter effettuare ritrasmissioni dei dati in caso di mancato ricevimento o di errore di trasmissione. Lo standard 802.15.4 definisce il livello 1 (livello fisico) e il livello 2 (livello MAC).

2.3.2 Componenti e topologie di rete di una WPAN

I dispositivi, dettate dalle specifiche IEEE 802.15.4, possono avere due diverse modalità: si parla di dispositivi *full function device(FFD)* e di *reduced function device (RFD)*. I primi possono agire come PAN coordinator, coordinator, o semplicemente Device. Gli RFD invece sono dispositivi estremamente semplici e utilizzano poche risorse, per questo possono rivestire solo il ruolo di device. Si ottiene una rete quando due o piú device all'interno dello spazio POS (Personal Operating Space) comunicano una rete è necessario avere almeno un dispositivo FFD che agisca come PAN coordinator. A seconda di quanto richiesto dall'applicazione le reti LR-WPAN possono operare secondo due topologie: a stella e peer to peer. Nella topologia a

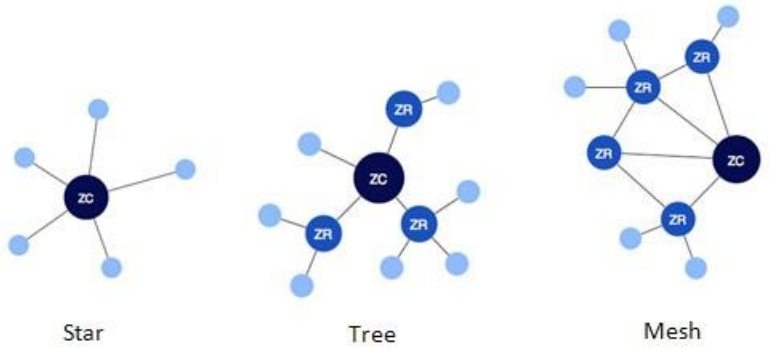


Figura 2.15: Topologie di rete

stella la comunicazione è stabilita fra un dispositivo e il PAN coordinator. Il device collegato ha tipicamente proprie applicazioni in esecuzione e rappresenta il punto di inizio o il termine della comunicazione. Il PAN coordinator puó , invece, iniziare o terminare la comunicazione ma anche instradare la comunicazione sulla rete. Tutti i device operanti su una rete sono dotati di indirizzi a 64 bit. Questo indirizzo puó essere usato per comunicazioni dirette all'interno della PAN, oppure puó essere rimpiazzato a discrezione del PAN coordinator dall'indirizzo breve quando il dispositivo si associa alla rete. Solitamente, viste le funzioni di rilevanza che riveste, il coordinatore è alimentato elettricamente e non da una batteria come accade invece per la maggioranza dei dispositivi semplici. La struttura di rete a stella si forma nel momento in cui un FFD si attiva e diventa PAN coordinator. Ogni topologia a stella è indipendente da qualunque altra risulti attiva, grazie all'identificativo scelto dal coordinatore che deve essere diverso da quelli già in uso dalle altre reti nel range di copertura. Non appena il PAN coordinator ha scelto il PAN identifier concede agli altri device nell'area (sia RFD che FFD) di associarsi alla rete.
Le applicazioni ideali per la topologia a stella sono: home automation, periferiche PC, giocattoli e cura personale.
Anche nella topologia peer to peer si ha un PAN coordinator, ma la differenza è rappresentata dalla possibilità dei device di comunicare fra loro direttamente. è evidente che questa topologia permetta la creazione di reti molto piú complesse. Puó prevedere l'instradamento multi-hop fra i vari device associati alla rete, funzioni di questo tipo esulano dallo standard 802.15.4 e sono lasciate al livello di rete. Una rete peer to peer si adatta ad applicazioni in campo di controllo e monitoraggio industriale, sensoristica, inventari e gestione magazzino, agricoltura, sicurezza, ecc. e puó essere ad-hoc, self-organizing oppure self-healing. Nella struttura di rete peer to peer ogni device è in grado di comunicare con qualunque altro device nella sfera di influenza. è eletto a PAN coordinator il primo dispositivo che comunica sul canale. Un esempio di questa topologia è rappresentato dalla rete cluster tree. Qui la maggioranza dei nodi sono FFD e i RFD si connettono come foglie alla fine dei vari rami, poichè ad essi è consentita una sola associazione per volta. Ogni FFD puó agire come coordinatore e fornisce il servizio di sincronizzazione agli altri dispositivi e agli altri coordinatori. Il PAN coordinator da origine al primo cluster della rete scegliendo l'identificativo univoco e inviando un beacon frame in broadcast ai dispositivi nelle vicinanze. Due dispositivi potrebbero tentare di stabilire una connessione come PAN coordinator nello stesso momento, per questo è necessario prevedere un meccanismo di risoluzione della contesa, ma esula dagli scopi dello standard. Il beacon frame, inviato dal coordinatore come richiesta di associazione alla rete, viene ricevuto dai device candidati a farne parte: se il PAN coordinator concede al dispositivo di unirsi viene aggiunto come figlio nella sua neighbour list. A sua volta il dispositivo appena aggiunto setta il

PAN coordinator come padre e lo aggiunge alla sua lista iniziando la trasmissione di beacon periodici. Se un dispositivo non è in grado di collegarsi al PAN coordinator cercherà un altro dispositivo padre nell'area. La forma piú semplice di cluster tree è dato dal cluster singolo, ma reti molto piú ampie e complesse possono essere ottenute tramite mesh di cluster multipli vicini. Il vantaggio di una topologia di questo tipo è evidente: permette un aumento notevole dell'area di copertura, anche se ció comporta un aumento del tempo di latenza dei messaggi.

2.3.3 Livello fisico

I servizi, che il livello fisico offre, intervengono sull'accensione e sullo spegnimento del modulo radio. Questa opzione permette al sistema il risparmio d'energia ed effettua, inoltre, la scelta del canale di comunicazione calcolando una stima sull'occupazione del canale ed analizza e calcola il rapporto segnale/rumore di un canale per scegliere il migliore diminuendo la probabilità di errori in trasmissione. Ovviamente modula e demodula i segnali in ricezione e in trasmissione.
Il livello fisico opera nella banda ISM utilizzando tre possibili bande libere: (Figura 2.16)

- 868-868.6 Mhz: banda utilizzata nella maggior parte dei paesi europei con un data-rate di 20kbps disponendo di un solo canale a disposizione;
- 902-928 Mhz : banda utilizzata nel continente oceanico e nel continente americano, offrendo un data-rate di 40Kbps e 10 canali disponibili;
- 2400-2483.5 MHz: banda utilizzabile in quasi tutto il globo con un data-rate massimo di 250 Kbps e 16 canali a disposizione.

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Figura 2.16: Bande di frequenza per le WSN

La modulazione del segnale piú diffusa è il DSSS utilizzando tecniche BPSK, O-QPSK, anche se recentemente sono state introdotte nuove modulazioni in

aggiunta al DSSS come PSSS. Il datagram a livello fisico, come mostrato Figura 2.17, è formato da una serie di campi: all'inizio si trova il preambolo, formato da 32 bit, con finalità di sincronizzazione tra i nodi. Successivamente viene utilizzato un otteto (11100101) prefissato che funge da indicatore di inizio pacchetto. Segue un campo physical header che indica la lunghezza del payload. Il payload è chiamato Phisical Service Data Units (PSDU) che puó avere una lunghezza massima di 127 bytes.

Figura 2.17: Datagram a livello fisico

2.3.4 Livello MAC

Il secondo livello (MAC) offre servizi quali la possibilità di creare una rete PAN, la trasmissione dei beacon e l'accesso al canale tramite il protocollo CSMA/CA. Questo livello supporta algoritmi di cifratura basata su AES-128 per la sicurezza dei dati, gestisce inoltre l'handshake, cioè gli Acknowledge per la ritrasmissione dei dati in caso di mancata o erronea ricezione. Calcola e verifica l'integritá della PDU. Puó supportare reti fino ad un massimo di 65536 nodi poichè utilizza un indirizzamento fino a 16 bit.
Il livello MAC prevede una struttura chiamata superframe. Questa è costruita dal coordinatore della rete ed è contenuta tra due messaggi chiamati beacon. I beacon contengono informazioni che possono essere utilizzate per la sincronizzazione dei dispositivi, per l'identificazione della rete, per descrivere la struttura della superframe stessa e la periodicità di spedizione dei beacon. La superframe è divisa in 16 slot temporali di uguale grandezza, dove il beacon frame è trasmesso nel primo e nell'ultimo slot di ogni superframe. Si possono avere due tipi di superframe: senza GTS (Guaranteed Time Slot) e con GTS. Quando viene inviata una superframe senza GTS, l'accesso al canale è regolarizzato dal protocollo CSMA/CA dove ogni dispositivo deve competere con gli altri per assicurarsi l'accesso ad uno slot. Questo periodo è chiamato CAP (Contention Access Period), visibile nella Figura 2.18.

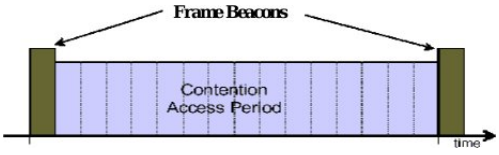


Figura 2.18: Superframe senza GTS

In altri tipi di applicazioni, invece, si necessita di costruire reti tali da ga-

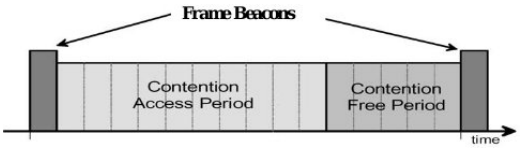


Figura 2.19: Superframe con GTS

rantire a tutti i nodi di poter trasmettere. è il caso di superframe con GTS, il quale dedica fino ad un massimo di sette slot temporali, detti CFP (Contention-Free Period), a determinati nodi. In questo periodo possono comunicare solamente i nodi prestabiliti e l'accesso al canale non è piú CSMA/CA (Figura 2.19).

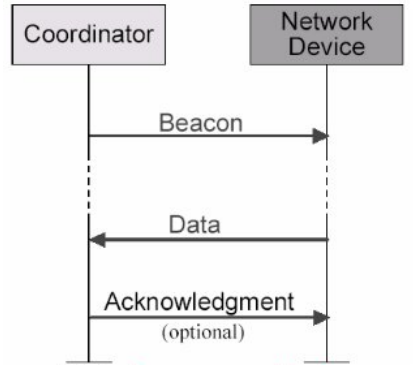
Modalità di trasferimento

Esistono tre diversi tipi di trasferimento dati: da un dispositivo al coordinatore (data transfer to a coordinator) dal coordinatore ad un dispositivo (data transfer from a coordinator) da un dispositivo ad un altro dispositivo (peer to peer data transfer). In una topologia di rete a stella si incontreranno solo i primi due tipi di trasferimento, mentre il terzo è tipico delle topologie peer to peer. Ogni tipo di trasferimento è regolato in maniera differente a seconda che la rete supporti la trasmissione dei beacon oppure no. L'abilitazione dei beacon è utile soprattutto in quei contesti in cui si ha low latency device, come il caso delle periferiche dei pc; nel caso in cui questi dispositivi non siano disponibili, si puó normalmente utilizzare una modalità con beacon disabilitati per trasferimenti normali, mentre continueranno ad essere utili e necessari al momento dell'associazione dei dispositivi alla rete.

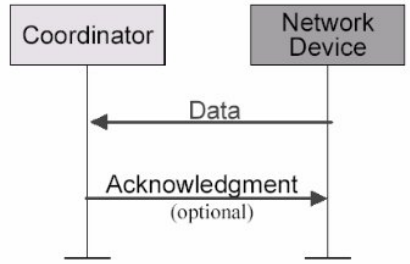
Data transfer to a coordinator:

- beacon enabled: In questa modalità, quando un dispositivo vuol trasferire dati al coordinatore, prima ascolta il canale in attesa del network beacon. Una volta ricevuto il beacon si sincronizza e quando è il momento trasmette utilizzando la modalità di accesso al canale slotted CSMA-CA. Una volta trasmesso il data frame il coordinatore che l'ha ricevuto puó inviare l'ack di conferma della ricezione. L'acknowledgment è opzionale e dipende dalle impostazioni della rete.
- beacon disabled: Se un device vuole trasmettere al coordinatore, nel caso in cui i beacon non siano abilitati, lo fa semplicemente senza ascoltare prima il canale, usando unslotted CSMA-CA.

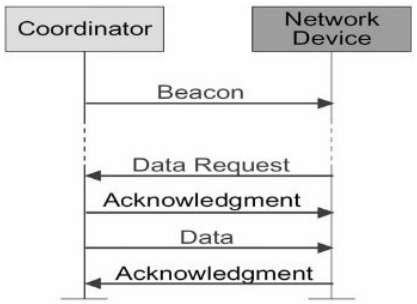
Data transfer from a coordinator:



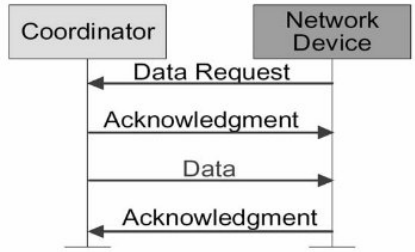
(a) da nodo a coordinatore con Beacon abilitato



(b) da nodo a coordinatore con Beacon disabilitato



(c) da coordinatore a nodo con Beacon abilitato



(d) da coordinatore a nodo con Beacon disabilitato

Figura 2.20: Trasmissione con beacon abilitati e beacon disabilitati

- beacon enabled: Il coordinatore che vuole trasmettere a un dispositivo in una rete con beacon abilitati trasmette un beacon in cui indica che esistono messaggi in sospeso. I device ascoltano periodicamente il canale in attesa dei beacon e leggendo che esistono messaggi in sospeso inviano una richiesta MAC (MAC command) usando slotted CSMA- CA per richiedere il pacchetto dati. Il data frame in attesa viene spedito a sua volta utilizzando slotted CSMA- con beacon abilitati CA, è opzionale l'invio di un ack di conferma ricezione dal device al suo ricevimento. La transazione ora è completa e l'indicazione di messaggi in sospeso viene rimossa dal beacon.

- beacon disabled: Se un coordinator ha dati da spedire invece di comunicarlo tramite un beacon immagazzina il data frame pendente. Un device puó contattare il coordinatore inviandogli un MAC command per richiedere eventuali pacchetti in attesa di invio, utilizzando unslotted CSMA-CA. A questo punto il coor-

dinatore invia un ack relativo al ricevimento della richiesta e successivamente invia un data frame. Nel caso non esistano dati in attesa di invio questo sarà un frame con payload a lunghezza zero. La trasmissione termina dopo la ricezione dell'ack inviato dal device al coordinatore per confermare di aver ricevuto il data frame.

Peer to peer data transfer:
In una PAN peer to peer ogni dispositivo puó comunicare con ogni altro nella sfera radio di copertura. Per poterlo fare i device che vogliono comunicare devono ricevere costantemente o sincronizzarsi. Se stanno ricevendo in modo costante, nel momento in cui vogliono trasmettere è sufficiente che lo facciano trasmettendo i loro data frame usando unslotted CSMA-CA. In caso contrario, ovvero in caso sia necessaria la sincronizzazione sono necessari altri provvedimenti che tuttavia non sono trattati dallo standard IEEE 802.15.4.

Struttura dei frame

La struttura dei frame è studiata appositamente per ridurre al minimo la complessità, ma al tempo stesso per garantire la robustezza della trasmissione su un canale che non essendo cablato è facilmente soggetto a interferenze. Ogni livello successivo dello stack ISO/OSI aggiunge rispetto a quelli qui trattati un header e footer rispettivamente a inizio e fine del frame. La specifica LR-WPAN distingue 4 tipologie di struttura:

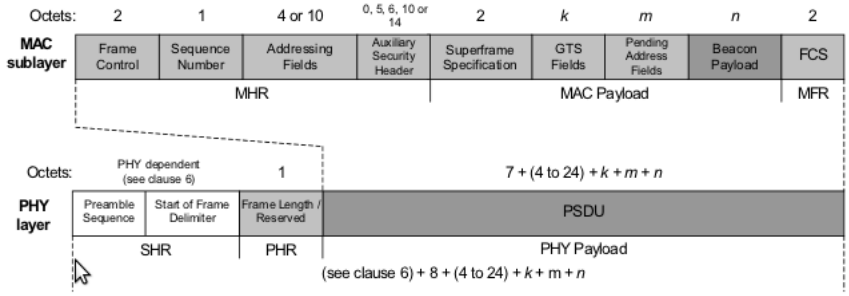


Figura 2.21: Vista schematica del beacon frame e del PHY packet

1. Beacon Frame
   La Figura 2.21 mostra la struttura del beacon frame. Questo frame è usato dal coordinator nella modalità beacon enabled per trasmettere i beacon. Il frame è suddiviso in tre parti: un' intestazione, un corpo centrale e una coda. Il corpo centrale è il MAC service data unit (MSDU) ed è formato da: specifica del superframe, specifica degli

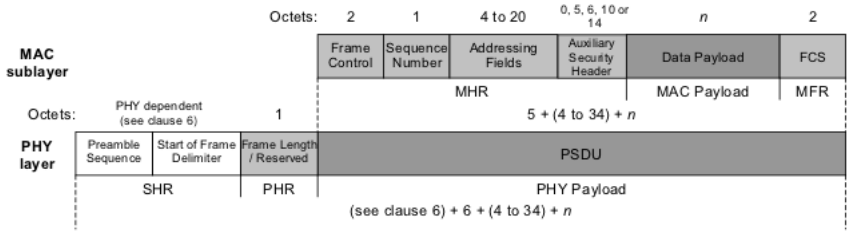


Figura 2.22: Vista schematica del data frame e del PHY packet

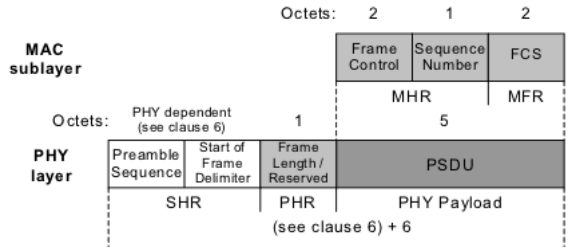


Figura 2.23: Vista schematica del acknowledgment frame e del PHY packet

indirizzi sospesi, lista indirizzi e campi beacon payload e GTS. L' intestazione, MAC header (MHR), contiene campi di controllo del frame MAC, il beacon sequence number (BSN) e campi indirizzo. La coda, il MAC footer (MFR) contiene 16bit frame check sequence (FCS). Tutte le tre parti unite formano il MAC beacon frame (MPDU). Il MAC beacon frame viene passato al livello fisico come PHY service data unit (PSDU). Il livello fisico aggiunge un'ulteriore intestazione, composta dal synchronization header (SHR) e dal PHY header (PHR). La prima contiene la preamble sequence, che serve ad abilitare il ricevente ad ottenere i simboli della sincronizzazione, e i campi start-of-frame delimiter (SFD). L' header del livello fisico (PHR), invece, contiene la lunghezza in byte del PSDU. Queste tre parti costituiscono il PHY beacon packet (PPDU).

2. Data frame
   La Figura 2.22 mostra la struttura del data frame. Il frame è diviso in tre parti: un' intestazione, un corpo centrale e una coda. Il corpo centrale è il MAC service data unit (MSDU) ed è formato dai dati, l' intestazione è il MAC header e contiene frame control, sequence number (DSN) e campi indirizzo, mentre la coda è il MAC footer e contiene 16 bit frame check sequence (FCS). Tutte le tre parti unite formano il MAC data frame (MPDU). Il MAC data frame viene passato al livello

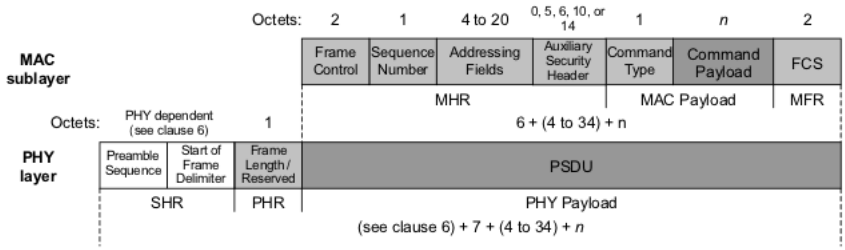


Figura 2.24: Vista schematica del MAC command frame e del PHY packet

fisico come PHY service data unit (PSDU) ovvero la parte centrale che contiene i dati. Il livello fisico aggiunge un'ulteriore intestazione, composta dal synchronization header (SHR) e dal PHY header (PHR). La prima contiene la preamble sequence, che serve ad abilitare il ricevente ad ottenere i simboli della sincronizzazione, e i campi start-of-frame delimiter (SFD). L' header (PHR) del livello fisico, invece, contiene la lunghezza in byte del PSDU. Queste tre parti costituiscono il PHY data packet (PPDU).

3. Acknowledgment frame
   La Figura 2.23 mostra la struttura dell'acknowledgment frame che viene creato dal livello MAC. E' formato unicamente da intestazione (MHR) e coda (MFR), non esiste la parte centrale contenente dati. L' header contiene i campi di controllo propri dei frame MAC e DSN (data sequence number). Il footer è composto come nelle altre tipologie di frame da 16 bit frame check sequence (FCS). Le due parti unite formano il MAC acknowledgment frame (MPDU). Il MAC acknowledgment frame viene passato al livello fisico come PHY service data unit (PSDU) ovvero la parte centrale che contiene i dati. Il livello fisico aggiunge un' ulteriore intestazione, composta dal synchronization header (SHR) e dal PHY header (PHR). La prima contiene la preamble sequence, che serve ad abilitare il ricevente ad ottenere i simboli della sincronizzazione, e i campi start-of-frame delimiter (SFD). L' header del livello fisico, invece, contiene la lunghezza in byte del PSDU. Queste tre parti costituiscono il PHY data packet (PPDU).

4. MAC command frame
   La Figura 2.24 mostra la struttura del MAC command frame che viene creato dal livello MAC. La parte data contiene il tipo di comando MAC. Al payload sono aggiunti come nelle altre strutture di frame un'intestazione (MHR) e coda (MFR). L' header contiene i campi di controllo propri dei frame MAC e DSN, i campi indirizzo ed eventualmente intestazioni ausiliarie di sicurezza. Il footer è composto come

nelle altre tipologie di frame da 16 bit frame check sequence (FCS). MAC payload, MHR e MFR formano il MAC acknowledgment frame (MPDU). Il MAC command frame viene passato al livello fisico come PHY service data unit (PSDU) ovvero la parte centrale che contiene i dati (payload). Il livello fisico aggiunge un' ulteriore intestazione, composta dal synchronization header (SHR) e dal PHY header (PHR). La prima contiene la preamble sequence, che serve ad abilitare il ricevente ad ottenere i simboli della sincronizzazione, e i campi start-of-frame delimiter (SFD). L' header del livello fisico, invece, contiene la lunghezza in byte del PSDU. Queste tre parti costituiscono il PHY data packet (PPDU).

2.3.5 Zigbee

Zigbee è il nome di una specifica per una suite di protocolli di comunicazione di alto livello, basati sullo standard IEE 802.15.4, nell'ambito delle WPAN, wireless personal area network. Le frequenze su cui opera maggiormente sono quelle assegnate a scopi industriali, specifici e medici: 2.4 GHz su scala mondiale, 868 MHz per l'Europa e 915 MHz per gli USA. Come descritto in Figura 2.25, Zigbee si inserisce nel livello network e nel livello applicazione che non vengono definiti nello standard 802.15.4 . Viene definita a basso costo perché puó essere utilizzata in svariate applicazioni e a basso consumo perchè offre alte prestazioni sulla conservazione delle batterie.

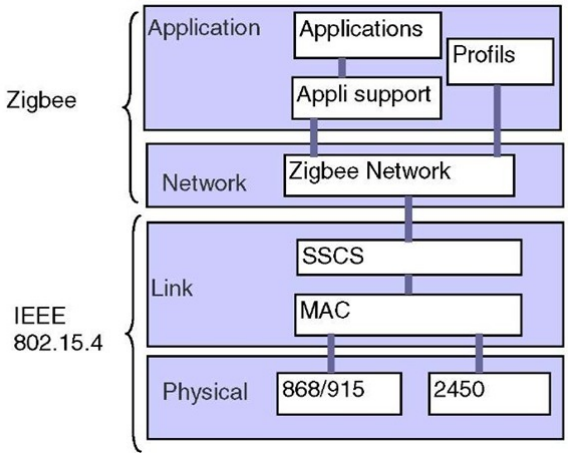


Figura 2.25: Il protocollo Zigbee

Livello Network

Le funzioni ZigBee del livello Network riguardano:

- Esecuzione dei comandi MAC come livello superiore;

- Individuazione delle figure di rete coordinatore, router ed end-device, ognuna con funzioni e compiti man mano decrescenti all'interno della rete;

- Auto-formazione e gestione delle connessioni di rete;

- Supporto dei pacchetti data e comando;

- Possibilità d'implementazione di un "Trust Centre" che gestisca le mansioni di sicurezza dell'intera rete ZigBee;

- Gestione delle chiavi network create con algoritmo di cifratura AES-128;

- Assegnazione di indirizzi a 16 bit per l'utilizzo nelle politiche di routing;

- Realizzazione dell'instradamento dei pacchetti attraverso due tecniche:

    - Tree Routing: i pacchetti destinati ad altri dispositivi nella rete vengono inviati al "padre" o al "figlio" secondo le associazioni effettuate con le primitive IEEE 802.15.4
    - Table Routing: un ciclo di "discovery" consente di conoscere quali sensori sono stati raggiunti da un pacchetto inviato in broadcast. Le risposte dei singoli dispositivi (pesati a seconda della funzione prescelta) consentono di realizzare ad ogni hop la tabella di routing.

Livello Applicazione

A livello applicativo ZigBee prevede l'identificazione di tre differenti strutture: Application Support Sub-layer (APS), Application Framework (AF), ZigBee Device Objects (ZDO). L'Application Support Sub-layer fornisce l'interfaccia fra livello di rete e livello applicazione (APL), applica o rimuove il livello di sicurezza e supporta i pacchetti data, comando e ACK.
L'Application Framework supporta i pacchetti di tipo KVP (Key Value Pair) e messaggio (MSG) e contiene gli oggetti applicativi con end-point da 1 a 240 (quelli utente) .
Lo ZigBee Device Objects si comporta come un'applicazione vera e propria, supportando funzioni classiche come il rilevamento di nuovi dispositivi e la gestione del nodo stesso. Possiede end-point 0, gestisce in modo intelligente le primitive a livello network, permette di effettuare discovery e binding, fornisce un'interfaccia di servizi fra oggetti applicativi e APS, inizializza APS, NWK. A livello di sicurezza, fornisce anche per il livello applicativo chiavi di sicurezza basate sull'algoritmo di cifratura AES-128, realizzando una struttura verticale di controllo della sicurezza, tale da rendere la rete ZigBee difficilmente attaccabile.

2.4 Reti di campo, CAN

il Controller Area Network (CAN) è un protocollo di comunicazione seriale in grado di gestire con elevata efficienza sistemi di controllo distribuiti di tipo real-time con un elevato livello di sicurezza e di integrità dei dati trasmessi. Una rete CAN nella sua versione piú tradizionale [2] è composta da un bus lineare realizzato con una coppia intrecciata di fili (schermati o meno) e da un numero teoricamente infinito di interfacce (nodi) ad esso collegati e che per mezzo di esso dialogano tra di loro come in Figura 2.26. Le specifiche CAN forniscono una descrizione del protocollo limitatamente ad una parte dei due layer piú bassi del modello ISO/OSI e cioè il Data Link ed il Physical Layer.
L'architettura CAN (Controller Area Network) è stata sviluppata da Bosh nel 1986 per Mercedes. Lo standard prevede un livello fisico e un livello

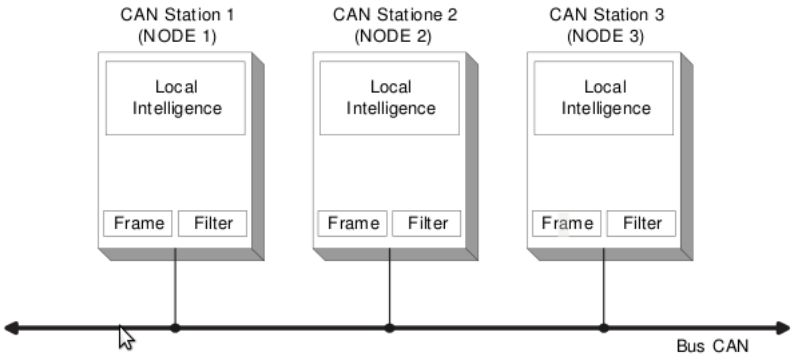


Figura 2.26: Struttura schematica di una rete CAN

datalink. Nel livello fisico si parla di cablaggi e la standardizzazione relativa al cavo e ai livelli di tensione. Il livello datalink si occupa della politica di accesso al canale, visto che è un canale condiviso, dell'indirizzamento, di definire i tipi e i formati dei pacchetti, e infine della rilevazione degli errori. Sopra questi c'è un livello applicazione chiamato CANOpen.

[2] Le specifiche del protocollo CAN non si occupano della descrizione del mezzo fisico, per sopperire a questa esigenza sono nati diversi standard ognuno dei quali con proprie peculiarità ed in grado di soddisfare particolari esigenze applicative. Uno degli elementi distintivi tra uno standard e l'altro è il mezzo trasmissivo impiegato per la realizzazione fisica della linea: coppie intrecciate di conduttori, bus a singolo conduttore, fibre ottiche, sistemi wireless via radio e infrarossi ecc.
In ogni caso si tratta di un bus di tipo lineare.

2.4.1 Livello fisico

Il mezzo trasmissivo è un doppino intrecciato, ovvero una coppia di conduttori isolati e attorcigliati tra di loro. La schermatura che ci potrebbe essere è una guaina di plastica oppure una calza di alluminio. L'intreccio e la schermatura servono per ridurre le interferenze, sia quelle che riceve dall'esterno che quelle che genera verso l'esterno. Tutte le stazioni si agganciano sullo stesso cavo, ovvero a un bus seriale. Non è una topologia a stella, ma una topologia a bus, dove il cavo fa un anello. Il cavo l'impedenza (resistenza del cavo) di terminazione è di 120 Ohm. La lunghezza massima del cavo dipende dalla velocità di trasmissione scelta, al fine di permettere che tutti i nodi possano sentire un bit scritto prima che sia terminata la scrittura. Infatti qui il sistema è piú reattivo, perchè non si trasmette piú un frame. Piú il bitrate è elevato, piú la lunghezza del cavo è ridotta. Quindi la topologia fisica prevede un cavo a cui tutti i nodi sono attaccati. Ciascun nodo è attaccato al cavo due volte, una per trasmettere e l'altra per ricevere. La trasmissione è differenziale quindi si puó trasmettere un solo bit in parallelo, anche se i conduttori sono due. Con trasmissione differenziale si intende la trasmissione dello stesso segnale invertito su ciascuno dei due conduttori. è un tipo di trasmissione asincrona, cioè non c'è un continuo invio di bit, ma quando qualcuno sta per trasmettere serve un qualcosa che sincronizzi prima di iniziare (poicheè il clock è assente). Il livello di tensione su un conduttore dev'essere opposto al livello di tensione sull'altro conduttore, cioè dev'essere uguale in valore assoluto ma opposto di segno. Questa trasmissione viene chiamata anche trasmissione bilanciata. Si è scelta questo tipo di trasmissione per due motivi:

1. Un filo percosso da corrente elettrica continua genera un campo magnetico attorno a sè, se poi la corrente varia, anche il campo magnetico varia e quindi si parla di onda elettromagnetica. Quindi se ad un cavo faccio passare un altro cavo, quest'ultimo raccoglie le onde elettromagnetiche, e il campo magnetico indotto induce a sua volta corrente. Quando invece trasmetto su due cavi vicini,ciascuno con livelli di tensione opposti, l'uno annulla l'effetto magnetico dell'altro e quindi non generano corrente indotta. Questo quindi per evitare le interferenze verso l'esterno.

2. Se arrivano interferenze da fuori, con una trasmissione sbilanciata, si creano picchi di tensioni sul cavo che potrebbero corrompere il valore dei bit. Invece con la trasmissione bilanciata, se crea interferenza su un conduttore la crea anche per l'altro e quindi gli errori si annullano.

Tutti i nodi possono scrivere sul canale, quindi nasce il problema di sovrapposizione di bit. Si implemento un meccanismo di wired AND: tutti possono scrivere sul canale ma lo zero è lo stato dominante e l'uno è il recessivo.

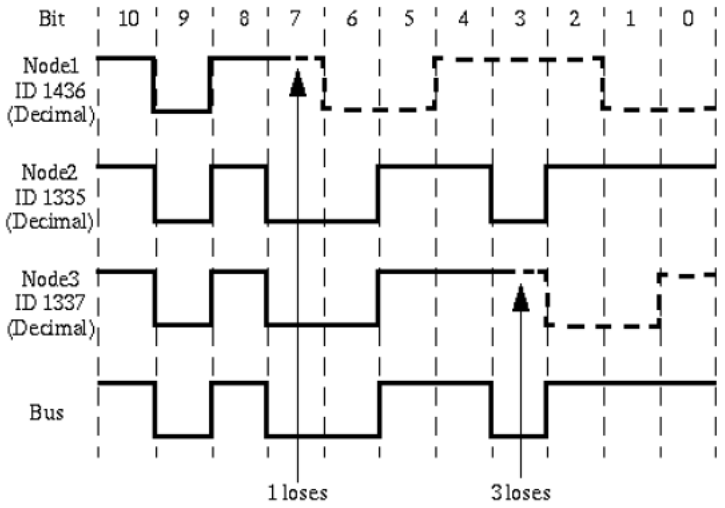


Figura 2.27: Meccanismo di WIRED AND

2.4.2 Livello Datalink

La politica di accesso al canale è un CSMA/CD modificato , rispetto a quello di Ethernet, e si basa proprio sul wired AND. Chi deve trasmettere ascolta prima il canale, se il canale è occupato aspetta che si liberi, se il canale è libero allora inizia a trasmettere. Puó succedere che due stazioni inizino a trasmettere nello stesso istante, ma è qui che entra in gioco l'utilità del wired AND. Durante la trasmissione, la stazione continua ad ascoltare il canale: appena legge sul canale un bit diverso da quello scritto la stazione smette di trasmettere. In questo contesto quindi non c'è mai collisione.

Nella Figura 2.27 si vede che ciascun nodo inizia a scrivere il proprio ID (a 11 bit) partendo dalla posizione piú significativa. Scrivono tutti 1 quindi sul bus ci sarà scritto 1. I tre nodi stanno ascoltando il canale, sentono lo stesso bit che hanno trasmesso e quindi non interrompono la trasmissione. Stessa cosa vale per il secondo e il terzo bit perchè tutti quanti scrivono 0 e poi 1. Quando trasmettono il quarto bit il primo nodo trasmette 1 mentre gli altri due trasmettono 0, e quindi, secondo il wired AND, il primo nodo perde la possibilità di trasmettere. Il secondo e il terzo nodo continuano a trasmettere senza problemi i successivi 0, 1 e 1. Il terzo nodo smette peró di trasmettere quando deve mandare come ottavo bit un 1 e nello stesso momento il secondo trasmette uno 0. Il secondo nodo quindi ottiene il possesso del canale e gli altri nodi, sentendo il canale occupato, aspettano che questo termini.

Si crea in questo modo un meccanismo di priorità, cioè chi ha l'id piú basso, tra quelli che vogliono parlare, vince.

Con questa idea hanno risparmiato nell'architettura , hanno risolto il problema della contesa e hanno anche creato un meccanismo di priorità. Un'ul-

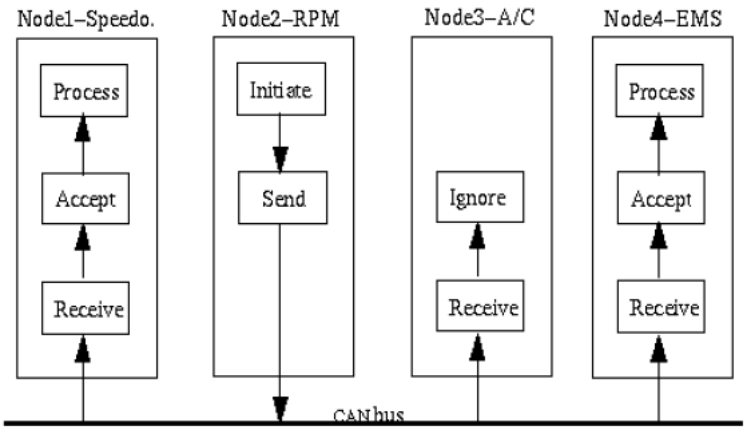


Figura 2.28: Esempio di trasmissione multicast

teriore vantaggio è l'assenza di tempi morti dovuti ad attese casuali che aumentano la latenza e introducono non determinismo.
L'indirizzamento non esiste.
Gli indirizzi non esistono intesi come mittenti e destinatari, ovvero l'identificatore non è relativo al nodo ma è relativo al dato. Piú stazioni possono trasmettere uno stesso tipo di dato e quindi potranno avere Id uguali. Il ricevitore , al termine degli 11 bit, capisce subito se è interessato o meno a leggere il dato . L'approccio infatti è multicast: la trasmissione non avviene per mittente e destinatario ma avviene basandosi sul concetto di gruppo, cioè chi scrive una cosa, la manda a tutti quelli che sono interessati a quel tipo di dato, perchè questi saranno gli unici che saranno rimasti ad ascoltare (Figura 2.28).

I vantaggi sono i seguenti:

- Nuove stazione possono essere aggiunte senza rischio di indirizzi duplicati. Se sono solo ricevitori non devo definire nuovi identificatori e quindi non serve modificare il SW degli altri nodi;
- Nessun meccanismo run time di assegnazione/gestione di indirizzi;
- Nessun frame duplicato in caso di trasmissione a piú destinatari;
- Non devo mandare pacchetti duplicati.

Gli svantaggi sono i seguenti:

- Non ci deve essere piú di 1 produttore per identificatore;
- Necessità di standardizzare gli identificatori. Bisogna creare una specie di dizionario.

2.4.3 Struttura dei frame

Il CAN definisce quattro tipi di messaggi detti anche frame:

Data Frame : è il messaggio di tipo piú comune ed è usato per trasmettere dati da un nodo ad un altro. Il frame inizia con un bit Start of frame usato per la sincronizzazione (hard synchronization) dei nodi e successivamente vengono trasmessi i campi[3] di seguito elencati.

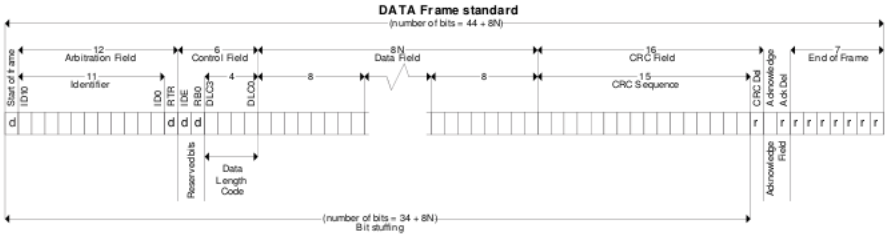


Figura 2.29: Formato del DATA Frame

- L'*Arbitration Field* contiene le informazioni necessarie durante la fase di arbitraggio per assegnare il possesso del bus quando questo viene conteso da piú nodi. Questo campo contiene l'identificatore (11 bit per un frame standard, 29 bit per un frame extended) e l'RTR (Remote Transmission Request) bit con cui si distingue un DATA Frame da un REMOTE Frame;
- Il *Control Field* è composto da 6 bit, gli ultimi 4 in ordine di trasmissione costituiscono il campo DLC (Data Length Code) che codifica il numero di byte di dato contenuti nel messaggio (0-8 byte). Se il frame è di tipo standard il primo bit del campo di controllo è il bit di IDE (Identifier Extention) che specifica proprio il tipo di formato (IDE=d standard, IDE=r extended), questa informazione se il frame è di tipo esteso è contenuta nell'Arbitration Field descritto in precedenza;
- Il campo *Data Field* contiene i byte che codificano l'informazione trasmessa con il messaggio. Il campo puó avere un'estensione che varia da 0 a 64 bit (solo multipli di 8) in base a quanto specificato nel campo DLC. I bit che compongono i singoli byte del campo vengono trasmessi partendo sempre dal piú significativo;
- Il *Cyclic Redundancy Field* (CRC Field) viene impiegato come sistema (non unico) per la rilevazione degli errori di trasmissione

[3] I bit SRR (Substitute Remote Request), RB1 e RB0 (Reserved Bit) non vengono descritti perchè sono risevati dal protocollo ed hanno valore costante.

ed è composto dalla CRC sequence (15 bit calcolati sulla base dei precedenti campi del frame) e da un bit recessivo, il CRC Delimiter bit utilizzato per chiudere la trasmissione del CRC ed essere certi che il bus una volta trasmessi i 15 bit sia in stato recessivo e quindi pronto a gestire correttamente la fase di acknowledgment che segue;

- Il campo successivo è l'*Acknowledge Field*. Durante l'ACK Slot bit il nodo trasmittente invia un bit recessivo (r), qualunque nodo della rete abbia ricevuto il messaggio formattato in modo corretto risponde inviando un bit dominante (d) che sovrascrive il precedente. Il nodo trasmittente riceve in questo modo l'ack cioè la conferma dell'avvenuta corretta ricezione da parte di almeno un nodo e chiude la trasmissione con un bit recessivo che costituisce l'ACK Delimiter.
- Il DATA frame si chiude con il campo di *End Of Frame* (7 bit recessivi).

Remote Frame : Una delle caratteristiche del protocollo CAN è che un nodo della rete non solo puó trasmettere informazioni o rimanere in attesa di riceverne altre, puó anche chiedere informazioni ad altri nodi della rete presentando una domanda. Per mezzo del REMOTE frame il nodo trasmette una richiesta remota cioè una Remote Transmit Request (RTR). Un REMOTE frame ha la stessa struttura di un DATA

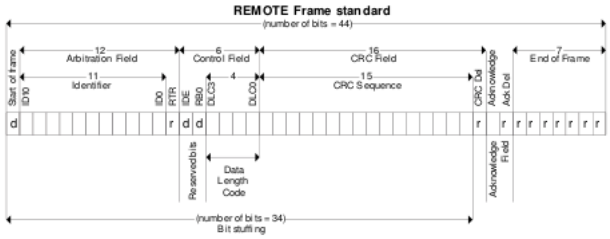


Figura 2.30: Formato del REMOTE Frame

frame con le seguenti differenze:

- il bit RTR è dominante per un DATA frame e recessivo per un REMOTE frame (per mezzo di esso il ricevitore riconosce il tipo di messaggio e si comporta di conseguenza);
- non esiste il campo DATA (non ha senso prevederlo);
- il campo DLC non codifica il numero di byte componenti il campo DATA ma puó essere impiegato per specificare altre informazioni: il tipo di dato o il numero di informazioni che il nodo richiede.

Error Frame : è generato da ogni nodo che rileva un errore[4]. Lo standard di CAN permette di tenere memoria dello stato del nodo. Ogni nodo ha due contatori di errore: uno di trasmissione e uno di ricezione. Sono entrambi inizializzati a 0. Il contatore di errore di ricezione viene incrementato di 1 alla volta, mentre quello di errore di trasmissione viene incrementato di 8 alla volta. Ogni stazione puó trovarsi in tre stati, come si vede alla figura Figura 2.31 . Si trova nello stato di er-

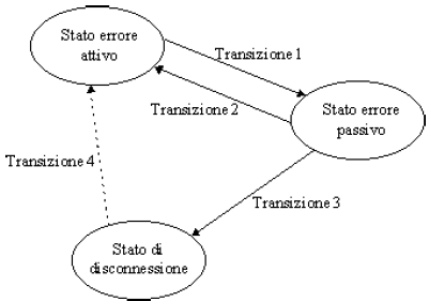


Figura 2.31: Stati d'errore

rore attivo (Error Active) se entrambi i contatori hanno valore minore di 127. In questo stato il nodo è nel pieno delle sue funzionalità e decrementa di 1 i contatori ogni volta che riceve un messaggio andato a buon fine. Quando rileva un errore spedisce un Error Frame costituito da 6 bit dominanti in modo da interrompere sempre la trasmissione.
Si trova nello stato di errore passivo (Error Passive) se uno dei due contatori ha valore tra 128 e 255. Il nodo è ancora in grado di eseguire tutte le sue funzioni, ma è probabile che esso presenti dei disturbi o condizioni di guasto. Quando esso rileva un errore spedisce un Error

[4]Gli errori cui il protocollo si riferisce, al rilevamento dei quali viene subito inviato un ERROR Frame, sono i seguenti:

- Bit Stuffing Error, al livello fisico i bit del frame vengono sottoposti al bit stuffing. Ovvero dopo 5 bit consecutivi di uno stesso valore viene inserito un bit di valore opposto.
- Bit Error, un nodo in trasmissione (che ha vinto la contesa) ascolta sempre il bus per verificare la corrispondenza con ció che sta trasmettendo: se esso ascolta un bit diverso dal suo (e non ci troviamo nell'Arbitration Field nè nell'Ack Slot) verrà segnalato un errore.
- Checksum Error, ogni nodo ricevente ricalcola il CRC in base a ció che ha ricevuto; se viene rilevato un *CRC Error* il frame viene inviato anche in questo caso ma solo successivamente all'ACK Delimiter.
- Frame Error, vengono vìolati alcuni campi fissi del pacchetto (bit che devono essere spediti sempre dello stesso tipo)
- Acknowledgement error, il trasmettitore non rileva alcun riscontro al frame appena inviato.

Frame di 6 bit recessivi che vengono interpretati come errore solo se nessuna stazione sta mandando un suo proprio messaggio; i bit recessivi vengono sovrascritti.
Si trova nello stato di disconnessione se uno dei due contatori ha valore maggiore o uguale a 256. Il nodo si stacca dal bus e non partecipa alla comunicazione lasciando agli altri nodi la possibilità di continuare a scambiarsi informazioni. Questo stato indica la presenza di un problema permanente che necessita di un intervento esterno per ripristinare il perfetto funzionamento. Alcune implementazioni consentono al nodo di tornare in Stato Errore Attivo dopo che aver ricevuto 128 occorrenze di 11 bit recessivi consecutivi (128 messaggi andati a buon fine) altre implementazioni invece necessitano di un reset hardware. Come mostrato in Figura 2.32, il frame di errore è composto da due

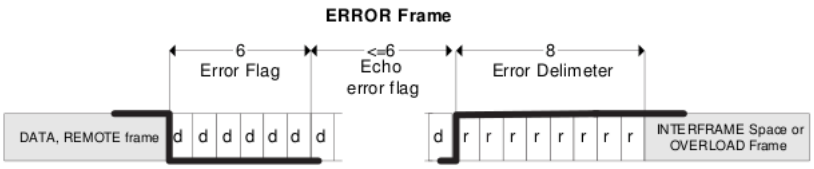


Figura 2.32: Struttura dell'ERROR Frame

campi chiamati Error Flag Field e Error Delimiter Field, quest'ultimo è composto da 8 bit recessivi a seguito dei quali i nodi possono riprendere le comunicazioni sul bus. La struttura del campo Error Flag dipende dallo stato in cui si trova il nodo al momento in cui il frame di errore deve essere generato. Se il nodo è Error Active, l'Error Flag è composto da 6 bit dominanti. Questa sequenza a seconda del momento in cui viene trasmessa vìola le regole di bit stuffing oppure la struttura del campo di Acknowledgement o ancora quella del campo di End Of Frame del messaggio che gli altri nodi della rete stanno decodificando. Indipendentemente da quando l'errore si verifica, l'ERROR Frame viola qualche regola di formato e come conseguenza anche gli altri nodi rilevano a loro volta un errore e trasmettono un ERROR Frame, i vari Error Flag si sovrappongono sul bus producendo un eco che ha come effetto quello di allungare la permanenza della sequenza di bit dominanti; l'Error Flag che ne deriva puó essere composto da un numero di bit dominanti che puó variare tra 6 e 12.
Se il nodo è Error Passive al rilevamento dell'errore il campo Error Flag prodotto è composto da 6 bit recessivi; il frame complessivamente è costituito da 14 bit recessivi. Come conseguenza l'ERROR Frame trasmesso da un nodo passivo non ha effetto sugli altri moduli della rete (l'informazione di errore trasmessa non è considerata attendibile perchè generata da una stazione CAN passiva quindi con qualche pro-

blema). Solo se il nodo Error Passive è in trasmissione ed esso stesso rileva l'errore, l'Error Flag di tipo passivo viola le regole di bit stuffing e quindi l'errore viene rilevato anche dagli altri nodi, solo in questo caso un ERROR Frame di tipo passivo ha effetto sulle altre stazioni della rete.

Overolad Frame : questo frame ha la stessa struttura di un ERROR Active Frame tuttavia puó essere generato solo durante un INTERFRAME Space e questa è anche l'unica maniera attraverso la quale i due messaggi possono essere distinti. Rispetto ad un ERROR Frame, l'OVERLOAD non incrementa i contatori e non causa la ritrasmissione dei messaggi, il messaggio ricevuto prima di un OVERLOAD Frame è un messaggio valido a differenza di quello che precede un ERROR Frame. L'OVERLOAD viene trasmesso al verificarsi di una delle seguenti

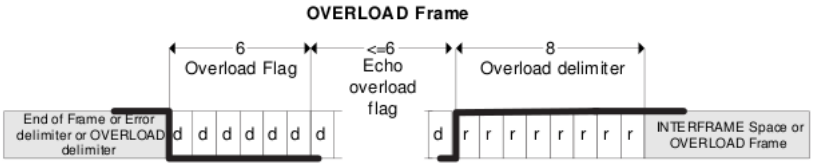


Figura 2.33: Strutura dell'OVERLOAD Frame

condizioni:

1. Il nodo per qualche motivo non è in grado di riabilitarsi in tempo per una nuova ricezione dopo che la precedente si è conclusa;
2. Viene rilevato un bit dominante durante la trasmissione dell'Intermission Frame; questa situazione non viene rilevata da alcuna delle condizioni di errore che sono successivamente descritte. Non determina la generazione di un Form Error e quindi tantomeno l'invio di un ERROR Frame. Se il bit dominante è l'ultimo (il terzo) viene interpretato come uno Start of Frame.
3. Viene generato anche se un nodo campiona un bit dominante come ultimo bit di un Error Delimiter o di un Overload Delimiter.
4. Se un nodo in ricezione rileva un bit dominate come ultimo bit di un End of Frame riconosce il messaggio come valido. Il bit dominante ricevuto è interpretato appartenente all'Intermission Frame e quindi il nodo si trova nella condizione descritta nel punto 1 (Figura 2.34, *d* sta per dominante e *r* per recessivo).

La struttura dell'OVERLOAD è analoga a quella dell'ERROR Frame e come tale la condizione di overload viene notificata a tutti i nodi della rete che a loro volta generano l'eco. L'obbiettivo che il nodo generatore

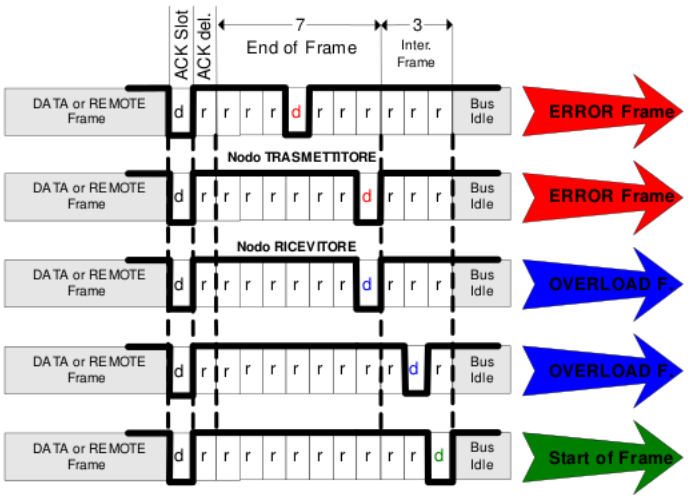


Figura 2.34: Comportamento del nodo CAN al rilevamento di errori nei campi di EOF e Intermission Space.

dell'OVERLOAD Frame raggiunge è quello di ritardare l'istante in cui il bus tornerà in condizione di idle e quindi sarà nuovamente disponibile a gestire altro traffico.

2.4.4 Tipi di nodi

Basic CAN

è l'implementazione piú economica: ha solo l'interfaccia di rete (va aggiunto microcontrollore) e un buffer di ricezione e di trasmissione gestiti con politica FIFO. Un messaggio puó essere ricevuto su un buffer mentre il microcontrollore sta leggendo su un altro buffer; se arriva un messaggio quando tutti i buffer sono pieni, esso viene scartato. Possono quindi andare perse delle informazioni nell'eventualità in cui il microcontrollore non sia abbastanza veloce. Un messaggio è inviato scrivendolo in un buffer di trasmissione. I messaggi a cui il nodo è interessato sono filtrati usando due registri che operano sull'identificatore del messaggio. Ogni bit della maschera puó essere settato ad 1, a 0 o Don't Care; la scelta va fatta molto accuratamente perchè ogni messaggio che supera tale controllo viene mandato al microcontrollore e il filtraggio finale verrà effettuato via software, generando carico di lavoro non necessario. Non supporta la risposta automatica ai Remote Frame e deve essere gestita dal software.

Full CAN

è l'implementazione piú costosa. Include un microcontrollore, un insieme di buffer chiamati mailbox a ciascuno dei quali viene assegnato l'identificatore a 11 bit e ad ogni tipo di messaggio compete il proprio buffer specifico. Quando viene ricevuto un messaggio vengono controllati tutti i buffer di ricezione; il filtraggio avviene interamente in hardware. In trasmissione, il messaggio viene memorizzato nel buffer che gli compete, viene attuata poi una politica di selezione del messaggio da trasmettere, favorendo quello con priorità piú alta. Quando viene ricevuto un Remote Frame, il nodo verifica se esiste un buffer di trasmissione con lo stesso identificatore: se si, il Data Frame corrispondente viene subito inviato senza chiamare in causa il microcontrollore. Rimane peró il rischio di inviare dati ormai vecchi che erano rimasti nel buffer.

2.4.5 I protocolli di alto livello del CAN

Gli elementi comuni ai diversi protocolli di alto livello del CAN possono sostanzialmente essere riassunti negli scopi stessi per cui nascono i protocolli di alto livello e cioè definiscono:

- il comportamento dei nodi e quindi della rete nella fase di start-up;
- le regole con cui assegnare gli identificatori dei messaggi ai vari nodi;
- il controllo del flusso dei messaggi, la sincronizzazione delle funzioni e la definizione di un riferimento temporale;
- la possibilità di avere messaggi in grado di trasportare piú dei tradizionali 8 byte di dati;
- il contenuto dei Data Frame cioè forniscono uno standard di codifica dell'informazione;
- i meccanismi di report dello stato del sistema (notifica degli errori e diagnostica);
- un modello per la descrizione delle funzionalità e dei parametri del dispositivo;
- un appropriato modello di comunicazione che normalmente puó essere di tipo Broadcast o Client-server.

Ogni HPL implementa queste funzionalità in modo diverso adattandole alle specifiche esigenze del campo applicativo per cui è stato sviluppato. CANopen, DeviceNet e SDS nascono per applicazioni industriali standardizzate e quindi abbastanza generali, CAL fornisce una base standard da impiegare per l'ottimizzazione delle applicazioni di rete, il SAE J1939 e l'OSEK/V-DX

sono stati sviluppati per essere impiegati nei sistemi di diagnostica nelle autovetture. In questo contesto verranno approfonditi solo i protocolli CAL e CANopen.

CAL

Il CAL, sviluppato da Philips Medical Systems, è il protocollo di alto livello adottato da un gruppo di produttori ed utilizzatori del CAN denominato CiA User Group. La peculiarità del CAL (CAN Application Layer) è quella di offrire un sistema di sviluppo per l'implementazione di reti distribuite basate sul CAN indipendente dall'applicazione da realizzare e consente un approccio object-oriented alla descrizione della stessa. Il CAL fornisce al progettista quattro distinti servizi di alto livello:

1. CMS (CAN-based Message Specification): per mezzo del quale vengono programmate le funzionalità dei singoli nodi. Il servizio CMS definisce otto classi di priorità per i messaggi, ogni classe dispone di 220 identificatori riservati.

2. NMT (Network ManagemenT): offre servizi di supporto per la gestione della rete come la sua inizializzazione, start e stop dei nodi, rilevazione di malfunzionamenti; il servizio è implementato sfruttando connessioni di tipo master-slave (esiste un solo NMT master).

3. DBT (DistriBuTor): offre una distribuzione dinamica degli identificatori CAN ai nodi della rete. Questi identificatori vengono chiamati Communication Object Identifier (COB-ID).

4. LMT (Layer ManagemenT): offre la possibilità di modificare i parametri di configurazione dei nodi come ad esempio l'indirizzo NMT di uno dei dispositivi o il bit timing e baudrate della rete.

CANopen

Il protocollo CAL mette a disposizione una completa serie di servizi per la gestione della rete e dei messaggi ma non definisce il contenuto degli oggetti CMS; sostanzialmente definisce come avviene avviene la comunicazione e come il protocollo CAN viene sfruttato ma non definisce cosa transita sul bus ed è in questo contesto che si inserisce il CANopen. CANopen ([9]) usa un sottoinsieme dei servizi definiti dal CAL per fornire l'implementazione di un sistema di controllo distribuito completo. CANopen è stato sviluppato dal CiA User Group17; nato inizialmente per il controllo di sistemi industriali è oggi impiegato anche per veicoli off-road, strumenti elettromedicali, e nella gestione dell'elettronica in campo navale e ferroviario. Questo protocollo sfrutta il concetto dei profile per garantire ai sistemisti la possibilità di far

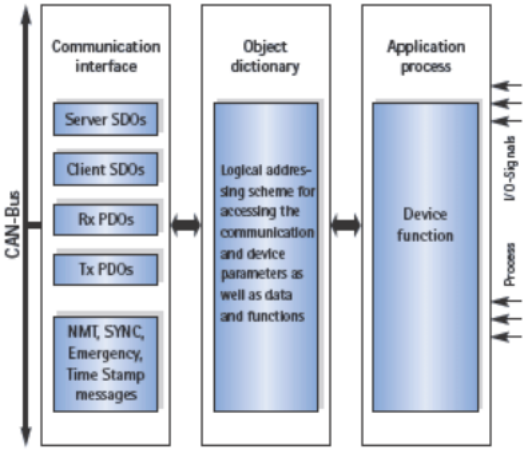


Figura 2.35: CANopen Object Dictionary

dialogare dispositivi realizzati da differenti produttori all'interno della solita rete senza l'esigenza di dover scrivere software dedicato per ogni singolo nodo. Per mezzo dei Device profile standard creati dal CiA le funzionalità piú comuni sono a disposizione del progettista, funzionalità piú specifiche e strettamente legate all'hardware dell'interfaccia vengono gestite attraverso profile messi a disposizione dai produttori stessi.

I profili sono descritti attraverso un database standardizzato detto Object Dictionary (OD). L'OD (Figura 2.35) è un concetto base dell'implementazione di CANopen mentre non è definito in CAL. Si tratta di una tabella in cui i vari gruppi di oggetti vengono identificati attraverso un indice a 16 bit e per consentire l'accesso ai singoli elementi della struttura dati viene utilizzato un ulteriore sottoindice a 8 bit. Per ogni nodo esiste un Object Dictionary che contiene tutti i parametri descrittivi del dispositivo ed il suo comportamento all'interno della rete. La parte dell'OD che descrive i parametri di comunicazione è comune a tutti i nodi, mentre una sezione è riservata alla caratterizzazione del dispositivo. Il Communication Profile è unico e descrive la forma generale dell'OD e gli oggetti che vengono riservati a contenere i parametri per la configurazione della comunicazione. Esistono invece piú Device Profile che definiscono in differente maniera alcuni degli oggetti dell'OD; ad ogni oggetto viene assegnato un nome, la funzione, indice e sottoindice (collocazione), tipo, viene specificato se è o meno obbligatorio e se l'informazione che contiene è modificabile o meno. I campi di questa tabella sono accessibili in lettura e scrittura ed i messaggi per mezzo dei quali avviene l'interrogazione e la risposta da parte dei nodi sono detti Service Data Object (SDO). Un SDO specifica attraverso il campo DATA del messsaggio CAN, indice e sottoindice dell'oggetto dell'OD che si vuole leggere o scrivere quindi attraverso questo tipo di messaggi il protocollo fornisce un

modo standard di accesso al dispositivo: segnali di I/O, parametri, funzioni e parametri di configurazione della rete.
Una rete CANopen deve avere un master e uno o piú nodi slave. Il master si occupa di gestire la sequenza di boot, manipola i campi dell'Object Dictionary e gli identificatori CAN dei vari dispositivi connessi. I dati possono essere trasmessi in modo sincrono consentendo di coordinare acquisizione dati e attuatori oppure possono essere inviati in ogni istante e consentono di notificare immediatamente ad un nodo una richiesta di trasferimento senza attenderne una sincrona. Il contenuto di entrambi i tipi di messaggi possono essere riconfigurati al boot della rete dal master durante la fase di network management. I 4 tipi di messaggi previsti dal protocollo sono i seguenti:

1. Administrative Message: sono i messaggi per la gestione della rete e la distribuzione degli identificatori. Sono messaggi ad elevata priorità che sfruttano i servizi NMT e DBT del CAL. Attraverso di essi al boot del sistema il master stabilisce un dialogo con ogni slave della rete e, stabilita la connessione, le informazioni di configurazione vengono scaricate sul nodo.

2. Special Function Objects: in questa categoria rientrano messaggi predefiniti tra cui:

   - SYNC : utilizzato per la sincronizzazione del trasferimento dei dati, ovviamente si tratta di un messaggio ad altissima priorità;
   - Time Stamp: fornisce un riferimento temporale comune a tutti i nodi;
   - Emergency: generato all'occorrenza per la segnalazione di uno o piú condizioni di errore;
   - Node/Life Guarding: il nodo che si occupa della gestione della rete (NMT master) monitora lo stato degli altri dispositivi. Ogni nodo viene continuamente interrogato per mezzo di un REMOTE frame, ogni nodo che supporta questa funzionalità risponde al master fornendo indicazioni sul proprio stato operativo. I timeout possono essere impostati in modo tale che se il master non riceve risposta entro un tempo prestabilito identifica il nodo come non funzionante e lo gestisce di conseguenza. CANopen prevede opzionalmente il controllo anche del nodo master da parte degli altri nodi e questa funzioanlità detta Life Guarding.
   - Boot-up: inviando questo messaggio l'NMT slave comunica al master il suo stato operativo. Ogni slaves funziona come una macchina a stati dotata di 4 step denominati Initializzation, Pre-operational, Operational e Prepared mode. Attraverso specifici messaggi, il master è in grado modificare lo stato di uno o piú nodi simultaneamente.

3. Process Data Object (PDO): sono utilizzati per il trasferimento real-time dei dati; le informazioni trasferite sono limitate a 8 byte e la natura dei dati è codificata attraverso l'identificatore CAN associato al messaggio (la tipologia del dato definisce anche la priorità del messaggio e quindi l' importanza dell'informazione trasmessa). La trasmissione puó avvenire in modo sincrono alla ricezione del messaggio SYNC che ha cadenza periodica oppure in modo asincrono su richiesta (a seguito della ricezione di un REMOTE frame) oppure al verificarsi di particolari eventi (se con il PDO si invia l'uscita digitale di un convertitore A/D usato per acquisire informazioni, quando il valore del segnale analogico cambia tramite un PDO l'interfaccia invia il valore aggiornato).

4. Service Data Object (SDO): attraverso gli SDO CANopen consente l'accesso ai campi dell'OD dei singoli dispositivi. Essendo impiegati in fase di boot per la configurazione della rete sono messaggi a bassa priorità che consentono il trasferimento di grosse quantità di dati (superiore agli 8 byte che costituiscono un limite invalicabile per i messaggi PDO).

Capitolo 3

Metodologie per sistemi embedded di rete

3.1 Progettazione dei Network Embedded System

I networked embedded system sono una particolare categoria di embedded system. Gli embedded system sono sistemi di eleborazione dedicati a scopi particolari; i NES sono embedded system che hanno una componente di comunicazione. Questa componente non è interna, perchè internamente è presente anche negli embedded system tradizionali (tipicamente il bus), ma è esterna. La parte di comunicazione verso l'esterno è decisiva per gli oggetti

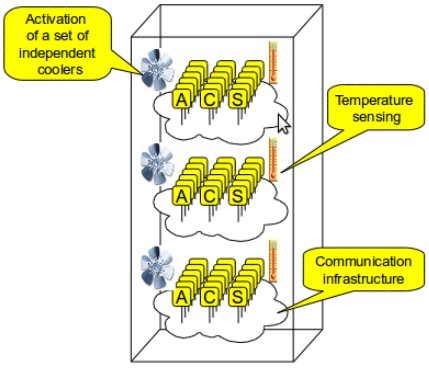


Figura 3.1: Controllo della temperatura di un edificio

di un NES. Le funzionalità di rete sono quindi un obbiettivo della progettazione e i requisiti di rete vengono dati assieme ai requisiti tradizionali. I sistemi embedded distribuiti sono gruppi di NES che sono collegati insieme usando interfacce di rete, protocolli standard e canali. Con queste applicazioni non interessa piú di tanto il comportamento di un singolo nodo ma

di tutti i nodi nel loro insieme, sul comportamento globale. Un esempio è quello del controllo della temperatura di un edificio descritto in Figura 3.1: ci sono tante stanze in cui ci sono tanti embedded system, rappresentati da scatolette. Ciascun dispositivo sente la temperatura in diversi luoghi dell'edificio e, per ogni stanza, in tanti punti, ciascuno dei quali comunica con gli altri e i controllori decidono se accendere o spegnere gli attuatori. Quindi servono tre tipi di dispositivi in rete:

A=attuation, accende o spenge;

S=sensing, legge la temperatura;

C=controller, decide se far accendere o spegnere in base ai dati ricevuti;

L'interesse finale è il riscaldamento dell'edificio: si tratta di un comportamento globale e non relativo al singolo nodo. Ci sono quindi centinaia di task eterogenei concorrenti montati su dispositivi di capacità diverse che sono collegati tra di loro tramite canali wireless o wired. Un altro aspetto critico è l'aspetto topologico: in un edificio non si possono mettere tutti i sensori in una stanza, bisogna distribuirli. Rimangono quindi delle domande da porsi: Quanti nodi mettere? Come assegnare i task a i nodi? Quali protocolli di rete usati? Quali sono i sistemi intermedi?
Quest'esempio è utile inoltre per definire i seguenti concetti:

requisito applicativo: come deve comportarsi il sistema;

requisito applicativo funzionale: deve rilevare la temperatura;

requisito applicativo non funzionale: deve consumare poco;

Partendo da questi requisiti si puo iniziare a costruire il sistema embedded. Il flusso tradizionale di un sistema embedded (Figura 3.2), descritto in Figura 3.2 è strutturato nel seguente modo:

1. Parte dai requisiti applicativi che possono essere funzionali o non funzionali. Attraverso un certo tipo di linguaggi i requisiti vengono modellati: questa fase viene chiamata Model Driven Design. Il modello ottenuto è un modello astratto.

2. Nella fase Design Space Exploration si parte dal modello astratto generato al passo precedente e lo si trasforma nel risultato reale finale; per farlo c'è bisogno anche della descrizione della piattaforma che verrà utilizzata. Nella Design Space Exploration si esplora lo spazio di progetto, ovvero si analizzano le varie scelte implementative che permettono di ottenere un risultato finale ottimo. Questa operazione richiede di prendere la descrizione delle funzionalità ottenuta nel modello astratto e mapparla su oggetti reali. Si parla di Platform Based Design (concetto inventato da Alberto Sangiovanni Vicentelli) per indicare un piattaforma come catalogo di oggetti reali.

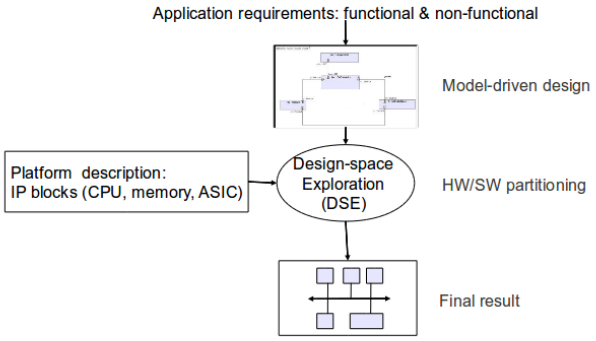


Figura 3.2: Flusso tradizionale di un Embedded System

3. Il modello astratto assieme a questo catalogo permettono di realizzare il risultato finale.

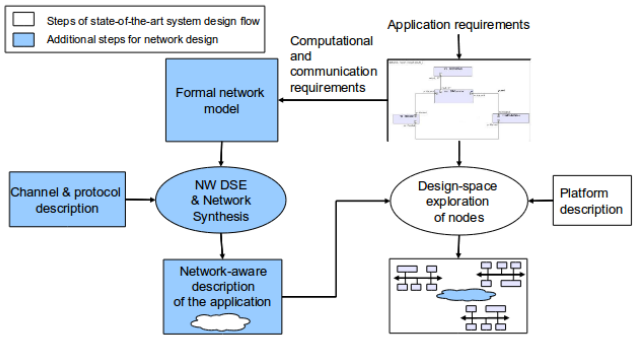


Figura 3.3: Flusso tradizionale di un Networked Embedded System

Il flusso di un Networked Embedded System estende il flusso tradizionale di un Embedded System perchè ha una componente in piú: la rete.
La Figura **??** rappresenta i passi che prevede il flusso:

1. Parte dai requisiti applicativi (funzionale e/o non funzionali) come nel flusso tradizionale;

2. Viene fatta una modellazione delle comunicazioni a partire dai requisiti computazionali e di comunicazione.

3. Si modella quindi una design space exploration sulla rete, che tiene conto di un modello iniziale dei requisiti di rete e di un catalogo di protocolli e di canali;

4. Si fa una modellazione model driven e si genera il modello astratto;

5. Come nel flusso tradizionale, nella fase Design Space Exploration si parte dal modello astratto generato al passo precedente e lo si trasforma nel risultato reale finale;

Si sceglie di fare prima la parte di rete e poi di progettare il nodo perchè iniziando con la parte di rete, la progettazione dell'hardware e del software risulta semplificata. Il viceversa sarebbe piú difficile.

La metodologia proposta per la progettazione dei sistemi embedded di rete è una metodologia che combina UML/MARTE, network synthesis e simulazione. La metodologia prevede:

- di mettere i modelli di UML/MARTE al centro del flusso di progettazione, in modo che non vengano usati solo all'inizio, ma che vengano raffinati progressivamente attraverso la simulazione e la design space exploration;

- di usare il nuovo sotto-profilo GQAM per rappresentare gli aspetti di computazione e di comunicazione del sistema;

- di modellare il risultato della network synthesis in UML/MARTE in modo che venga rappresentato non solo un singolo nodo, ma l'intero sistema distribuito: questo si ottiene usando GQAM;

- di affidarsi a i modelli eseguibili SystemC non solo per la simulazione funzionale ma anche perla simulazione di rete;

- di fornire i modelli eseguibili SystemC anche per ulteriori passi della progettazione;

La metodologia proposta è mostrata in Figura 3.4. I modelli UML/-MARTE sono al centro della progettazione. I modelli sono progressivamente costruiti e raffinati, usando i tre tool circostanti per la simulazione e la network sinthesis; le frecce sono usate per rappresentare il flusso delle informazioni tra UML/MARTE e questi tool; i numeri servono a definire la sequenza dei passi di progettazione. Gli input del flusso sono i requisiti applicativi, i vincoli ambientali, e la descrizione dei nodi e dei canali.
Il secondo passo viene denotato come *System View*, visto che l'enfasi è sulla funzionalitá e lo scambio di informazioni senza decidere se la comunicazione è interna a un singolo chip o implementata come rete esterna.
Il terzo passo, *Network Synthesis*, viene creato il sistema distribuito facendo

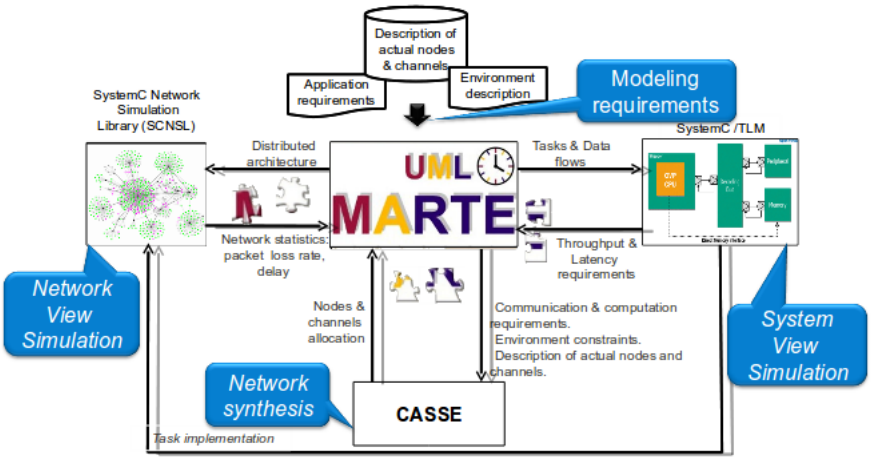


Figura 3.4: Design flow di un Networked Embedded System con UML MARTE

il mapping dei task nei nodi, dei data flow nei canali, e posizionando i nodi nell'ambiente.
Infine i nodi e i canali vengono modellati usando una libreria di SystemC, SCNSL, che riproduce il comportamento delle reti basate su pacchetti: questo ultimo step viene chiamato *Network View* visto che le funzionalitá del sistema vengono simulate in uno scenario di rete completo.

3.1.1 Modellazione dei requisiti

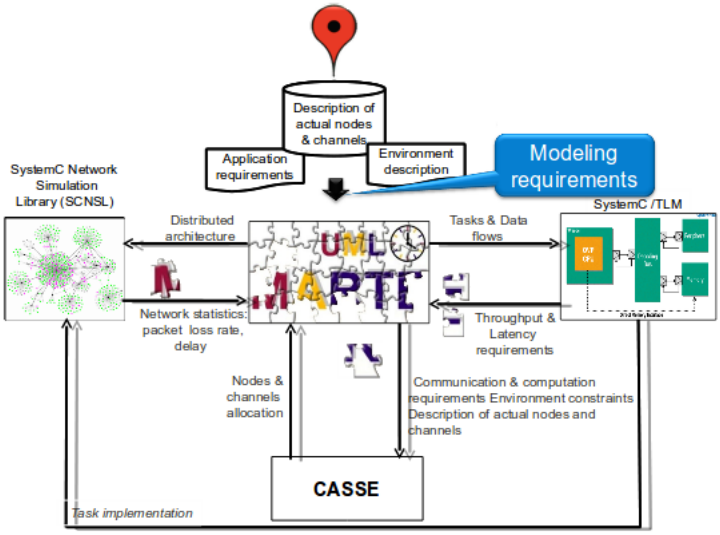


Figura 3.5: Modellazione dei requisiti

Un linguaggio di modellazione è un linguaggio artificiale che puó essere usato per esprimere informazioni, conoscenze o sistemi in una situazione che è definita da un insieme di regole consistenti. Le regole sono usate per l'interpretazione del significato dei componenti nella struttura.
Un linguaggio di modellazione puó essere grafico o testuale.
I modelli possono essere raffinati continuamente (come si vede dalla Figura 3.6) finchè l'applicazione è completamente specificata.
MARTE è un profilo di UML proposto da OMG per poter annotare i mo-

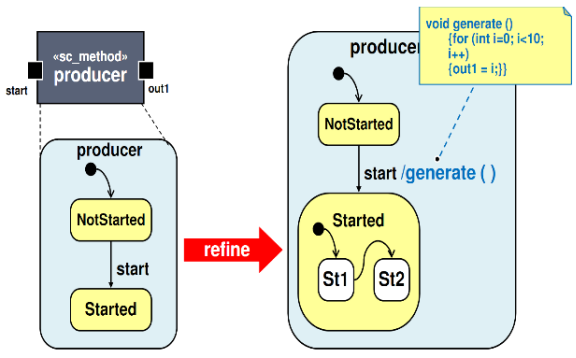


Figura 3.6: Raffinamento di un modello

delli con informazioni e vincoli di tipo non funzionali e real time.
Nella progettazione non rappresenta solo un punto di partenza ma anche un repository di tutte le versioni del progetto, che sono via via piú raffinate. MARTE consiste di un insieme di estensioni, molte delle quali si riferiscono ad aspetti non funzionali. I requisiti non funzionali di un'applicazione possono essere classificati in due categorie, aspetti quantitativi e aspetti qualitativi, e possono essere disponibili in diversi livelli di astrazione. Infine possono essere definiti per supportare la modellazione, l'analisi o entrambi. Per soddisfare tutti questi requisiti, MARTE è strutturato come una gerar-

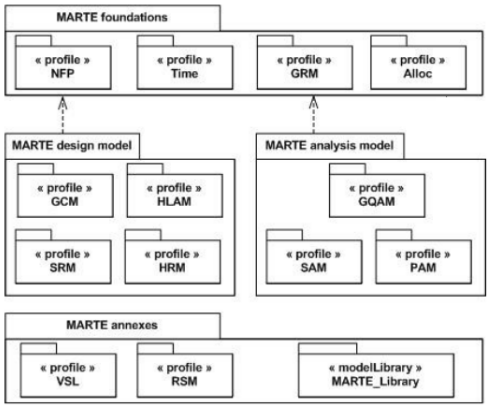


Figura 3.7: Descrizione dell'architettura di MARTE

chia di sotto-profili, come descritto in Figura 3.7 .
L'ultima versione di MARTE ha introdotto il Generic Quantitative Analysis Modeling (GQAM), un sotto-profilo che fornisce alcuni stereotipi interessanti per la rappresentazione degli aspetti di computazione e comunicazione dei sistemi embedded di rete:

- *GaExecHost* denota un processore che esegue dei passi;
- *GaCommHost* denota un link di comunicazione fisico;
- *GaCommChannel* denota una comunicazione logica che connette *SchedulableResources*

Per la modellazione dei requisiti gli aspetti fondamentali che devono essere rappresentati in UML/MARTE sono i task, i nodi, i canali e l'ambiente esterno. Come si vede in Figura 3.6, viene creata una classe per ciascuna di questi e i requisiti applicativi, funzionali e non funzionali, e i vincoli ambientali sono rappresentati attraverso la relazione tra queste classi. I tre stereotipi dal sottoprofilo GQAM sono usati in questo diagramma per specificare la semantica di alcune classi e dei suoi attributi.

1. Task: un task rappresenta una funzionalitá base dell'intera applicazione.

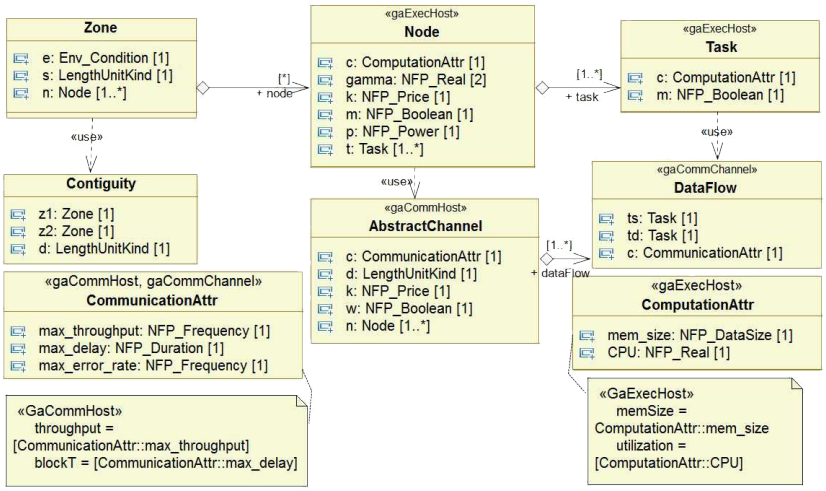


Figura 3.8: Entitá modellate come classi in UML MARTE

Prende alcuni dati in input e li fornisce in output. La classe *Task* è stereotipata come *GaExecHost*. L'attributo *c* rappresenta i requisiti di computazione, come l'utilizzo di memoria e CPU. Per questo attributo è stato creato e stereotipato come *GaExecHost* un nuovo tipo di dato *ComputationAttr*.
L'attributo *m* è un attributo booleano che rappresenta il possibile requisito di mobilitá del task.

2. Data Flow: un dataflow rappresenta la comunicazione tra due task; l'output di un task sorgente viene mandato come input ad un task destinazione. La classe *DataFlow* è stereotipata come *GaCommChannel* e ha come attributi il task sorgente *ts*, il task destinazione *td*, e i requisiti di comunicazione *c*.
   Per l'attributo *c* è stato creato un nuovo tipo di dato, *CommunicationAttr*, con tre campi: max_throughput, ammontare massimo di informazioni trasmesse nell'unitá di tempo max_delay, tempo massimo permesso per mandare il dato alla destinazione, max_error_rate, massimo numero di errori tollerati dalla destinazione. Questo tipo di dato è stereotipato come *GaCommHost*.

3. Nodo: un nodo è un contenitore di task. La classe *Node* è stereotipata come *GaExecHost* L'attributo *c* rappresenta le risorse disponibili sul nodo ed è dello stesso tipo dell'attributo *c* del task; il primo peró rappresenta le risorse di computazione fornite dal nodo, mentre il secondo rappresenta i requisiti di computazione di cui un task ha bisogno.
   p è la potenza del nodo, *k* è il costo del nodo, *m* è un attributo booleano che indica se il nodo puó essere mobile.

4. Canale Astratto: un canale astratto è una generalizzazione dei canali di rete, e contiene il canale fisico e tutti i protocolli. Un canale astratto connette i nodi.
   La classe *AbstractChannel* è stereotipata come *GaCommHost*.
   L'attributo n contiene un insieme di nodi che comunicano attraverso il canale astratto. L'attributo c rappresenta le risorse di comunicazione del canale astratto ed è dello stesso tipo dell'attributo c della classe *DataFlow*: il primo peró rappresenta le risorse di comunicazioni fornite dal canale astratto, il secondo invece rappresenta le risorse di comunicazione di cui il dataflow necessita. L'attributo d è la distanza massima dei nodi connessi da un canale astratto, k è il costo del nodo, w è un attributo booleano che indica se il canale è wireless.

5. Zona e Contiguitá: l'ambiente viene partizionato in zone che contengono uno o piú nodi e la contiguitá è la distanza tra le zone. Nella classe *Zone*, l'attributo n rappresenta l'insieme dei nodi contenuti in quella zona, s rappresenta le caratteristiche spaziali della zona, e rappresenta informazioni sull'ambiente. La classe *Contiguity* ha due attributi, *z1* e *z2* che rappresentano le zone coinvolte nella relazione, e un attributo d che rappresenta la distanza tra le due zone.

I vincoli a livello applicazione vengono dati invece dall'utente ad esempio: un'istanza del task T3 puó essere assegnata a ciascun nodo al massimo una volta. Questi sono specificati usando la cardinalità nella relazione tra gli oggetti. Sia i vincoli impliciti che quelli a livello applicazione vengono messi

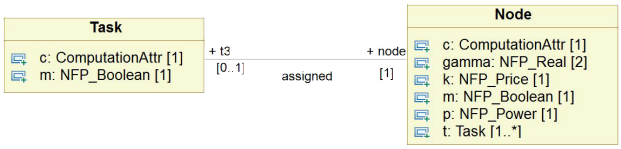


Figura 3.9: Classi Task e Nodo

tra i problemi di ottimizzazione della sintesi di rete.

3.1.2 System View Simulation

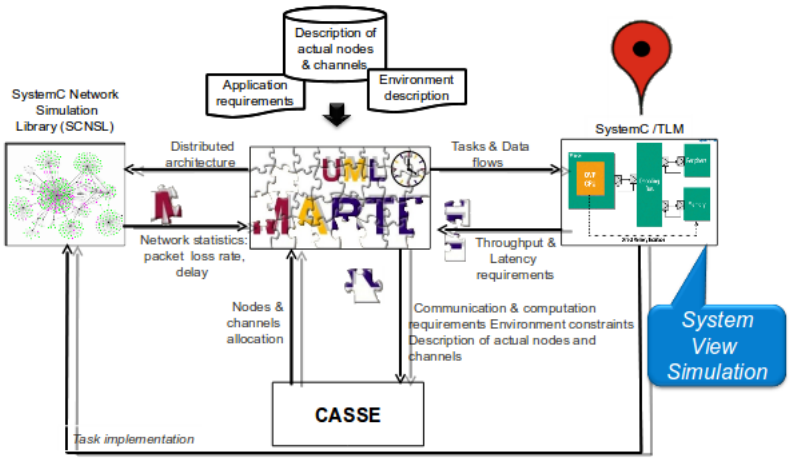


Figura 3.10: System View Simulation

Le descrizioni generate dal UML/MARTE in termini di task e dataflow sono usate per generare un modello eseguibile SystemC che riproduca il comportamento funzionale dell'applicazione.
Le istanze dei task sono modellate come *sc_modules* che implementano le interfacce *tlm::tlm_fw_transport_if* e *tlm: : tlm_bw_transport_if*.
I dataflow sono modellati usando i socket TLM. Dato un dataflow da un task t1 a un task t2, viene creata un'istanza di *tlm::tlm_initiator_socket* all'interno di t1 e un'istanza di *tlm::tlm_target_socket* all'interno di t2. Entrambe le istanze possono essere presenti nello stesso task se questo task è sorgente e destinazione di dataflow diversi. Queste regole permettono un mapping semi-automatico delle informazioni di UML/MARTE in SystemC/-TLM. L'esecuzione del modello SystemC permette di validare il comportamento funzionale dell'applicazione e di aggiustare i dettagli implementativi. È possibile inoltre stimare il throughput richiesto associato a ciascun dataflow. Il ritardo massimo ammesso puó essere migliorato partendo dalla specifica temporale di UML/MARTE e poi verificato attraverso l'esecuzione sincronizzata dei task nella simulazione. Queste informazioni poi vengono annotate in UML/MARTE e usate per la fase di Network Synthesis.

3.1.3 Network Synthesis

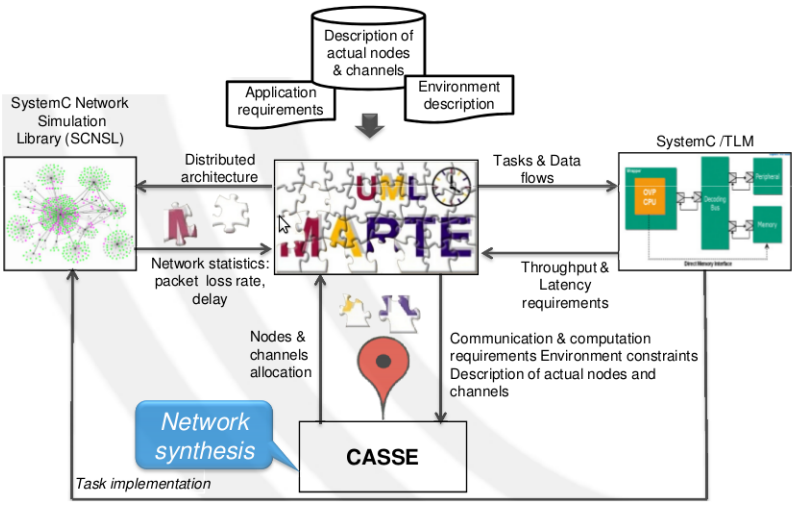


Figura 3.11: Network Synthesis

CASSE fornisce una notazione matematica per specificare le dimensioni della rete di un sistema embedded distribuito.
Tutte le informazioni sui vincoli, i requisiti di comunicazione, i canali e i nodi vengono estratti dal modello UML/MARTE e tradotti in una rappresentazione matematica.
CASSE permette di trovare un assegnamento ottimo dei task ai nodi, dei dataflow ai canali, e dei nodi alle zone, risolvendo un insieme di equazioni basate su questa rappresentazioni matematica. In Figura 3.12 viene mostrato questo passaggio.

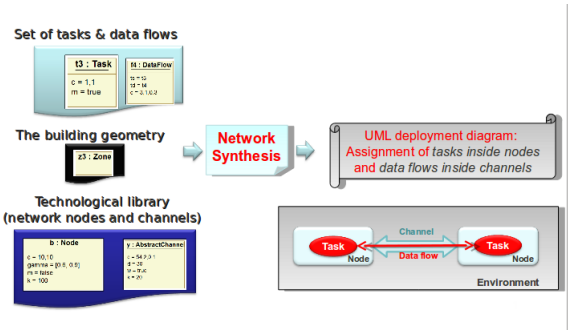


Figura 3.12: Traduzione delle informazioni sui vincoli,dei requisiti di comunicazione, dei canali e dei nodi in una rappresentazione matematica

Questo processo viene chiamato Network Synthesis e le sue soluzioni sono modellate usando il deployment diagram UML. Questo diagramma consiste di quattro tipi di elementi: pacchetti, risorse HW, risorse SW e i percorsi di comunicazione. In Figura 3.13 viene mostrato il deployment diagram per la soluzione del caso di studio.

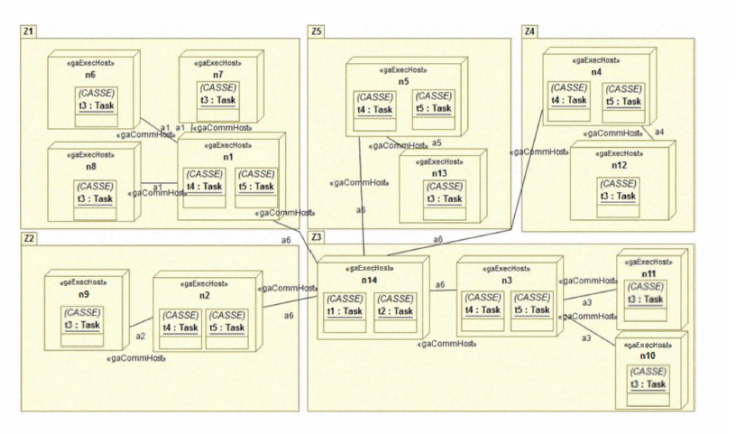


Figura 3.13: Deployment diagram per la soluzione del caso di studio

3.1.4 Network View Simulation

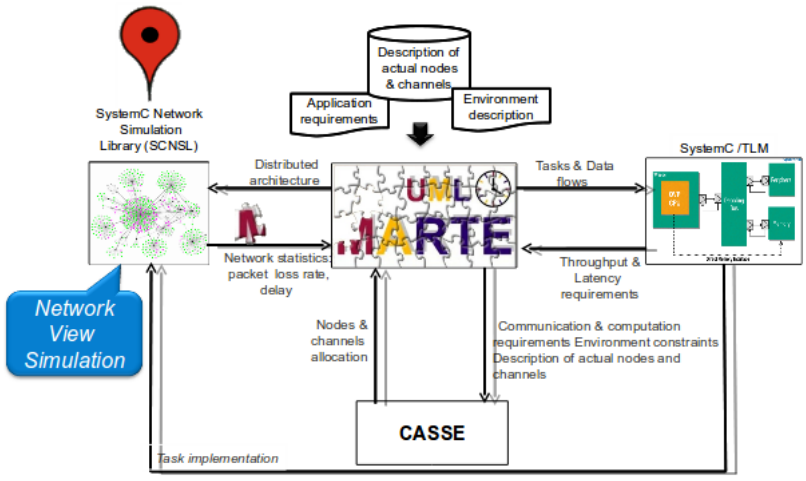


Figura 3.14: Network View Simulation

SystemC Simulation Network Library (SCNSL) è un'estensione del SystemC per consentire la modellazione di reti packet-based, quali le reti wireless, Ethernet, bus di campo, ecc. Come SystemC per i segnali sul bus, SCNSL fornisce primitive per modellare la trasmissione di pacchetti, la ricezione, la contesa sul canale e la perdita di canale wireless. L'utilizzo SCNSL permette la modellazione facile e completa di applicazioni distribuite in rete dei sistemi embedded, quali reti di sensori wireless, routing e controllo di sistemi (edifici, impianti per l'industria, ecc.).
I vantaggi che fornisce SCNSL sono:

- Semplicità: un unico strumento, cioé, SystemC, viene utilizzato per modellare sia il sistema (vale a dire, CPU, memoria, periferals) che la rete di comunicazione;

- Efficienza: la simulazione è piú veloce in quanto non è necessario un simulatore di rete esterna;

- Ri-uso: blocchi IP in SystemC possono essere riutilizzati;

- Scalabilità: supporto di vari livelli di astrazione nella descrizione dei dispositivi e del sistema;

- Software aperto: diversi strumenti disponibili per SystemC possono essere impiegati senza modifiche;

- Estensibilità: l'uso di standard SystemC e la disponibilità del codice sorgente garantisce l'estensibilità della libreria per incontrare vincoli progettuali specifici.

Nella fase di Network View, si parte dalla descrizione del sistema fornita da Marte e si produce un modello eseguibile SystemC con una rappresentazione esplicita della rete. L'implementazine dei task viene presa dalla seconda fase (System View); questi sc_modules vengono connessi ai nodi SCNSL con dei socket TLM. Durante la connessione dei task ai nodi, SCNSL fornisce anche la possibilitá di creare un path logico tra i task: questo meccanismo è usato per creare i dataflow. La trasmissione dei pacchetti e la ricezione puó essere a livello dei task cosí da ottenere statistiche sul packet loss rate e il delay. La simulazione della rete fornisce informazioni piú accurate sul comportamento di un applicazione distribuita; il max delay permette di effettuare un'analisi del caso peggiore, utile per scenari hard real-time. Questa informazione non puó essere derivata nè nella fase di System View Simulation nello Step 2, nè dal modello analitico nello Step 3.

Capitolo 4

Sistemi di controllo mediante rete

4.1 Descrizione

Un sistema di controllo è composto da un controllore e da un impianto che viene controllato. L'impianto è un sistema fisico. Il controllore invece è costruito dal progettista ed è un programma software che implementa una strategia di controllo. Il progettista costruisce anche i sensori, che leggono le informazioni dagli impianti, e gli attuatori, che agiscono sull'impianto. Il controllore, i sensori e gli attuatori sono blocchi digitali. Una rete basata su pacchetto trasporta sia i comandi da controllore a attuatore (forward) che le misure da sensore a controllore (backward). Questo porta sia scalabilità, perchè la rete puó essere condivisa da piú infrastrutture, che efficienza dei costi, perchè possono essere riusati i protocolli già esistenti. I Network Control System possono avere applicazioni per l'esplorazione dello spazio, l'esplorazione subaquea, la teleoperazione, ecc.

4.2 Problemi di controllo

La rete introduce dei nuovi problemi nel ciclo di controllo: ritardo di trasmissione, perdita di pacchetti e bit errors.

4.2.1 Ritardo di trasmissione

Il ritardo di trasmissione è il tempo totale impiegato tra la generazione dei comandi/misure e il suo utilizzo da parte degli attuatori/controllori. Contribuiscono al ritardo:

1. il packet encoding time, che dipende dalla dimensione del pacchetto e dalla velocità dell'interfaccia di rete;

2. il ritardo di propagazione, che dipende dalla velocità del segnale e dal range di trasmissione. è costante per le reti wired e variabile per le reti wireless.

3. il tempo di arbitraggio del canale, ovvero il tempo per ottenerne l'uso. è costante quando si usa una politica TDMA e variabile con il CSMA.

4. il tempo di ritrasmissione; l'interfaccia di rete aspetta un ack e potrebbe riprovare diverse volte prima di avere successo. Questo tempo è variabile e dipende dal livello di traffico e dalla qualità del canale.

5. il tempo che impiegano i sistemi intermedi (switch, router, access point, coordinatori) a processare le informazioni. Piú sono i sistemi intermedi da attraversare e piú alto è il ritardo.

Il valore del ritardo determina il ritardo del ciclo e dovrebbe essere compatibile con l'applicazione di controllo e quindi con i requisiti di reazione. La variazione del ritardo altera l'intervallo tra i pacchetti (frequenza di campionamento) e il ritardo del ciclo. Le assunzioni sulla frequenza di campionamento e il loop delay usate nel control design non sono soddisfatte a run-time.

4.2.2 La perdita dei pacchetti

Alcuni pacchetti possono non arrivare a destinazione. Le cause possono essere:

1. un guasto nel canale (non raro nelle reti wireless);

2. una queue overflow dovuta all'alto traffico;

3. il timeout del ricevitore (se un campione è troppo ritardato è inutile);

4. la detection degli errori sui bit.

Questo determina un ciclo di controllo temporaneamente rotto: i comandi non vengono applicati e le misure non vengono considerate dal controllore.

4.2.3 Errori sui bit

Uno o piú bit del pacchetto potrebbero essere sbagliati e questo evento non viene visto dall'error check. Le cause possono essere:

1. un guasto del canale;

2. l'interferenza delle altre sorgenti dei segnali;

3. la debolezza del segnale dovuta alla distanza e agli ostacoli.

Questo tipo di problema porta a dei comandi e/o delle misure alterati.

4.3 Soluzione 1: Servizi differenziati

DiffServ è una tecnica standard per introdurre la garanzia della Qualità del Servizio (QoS) nelle reti IP. In ciascun sistema intermedio ci sono due code per due diverse priorità di forwarding (cavi virtuali):

- coda H, per pacchetti ad alta priorità;
- coda L, per pacchetti a bassa priorità;

Lo scheduler assegna le priorità ai pacchetti impostando un campo appropriato nel loro header. Alla politica H dev'essere assegnato solo una frazione della banda totale altrimenti la differenziazione diventa inutile. L'architettura proposta è quella in Figura 4.1.

è composta a un sistema continuo (l'impianto), un sistema discreto con

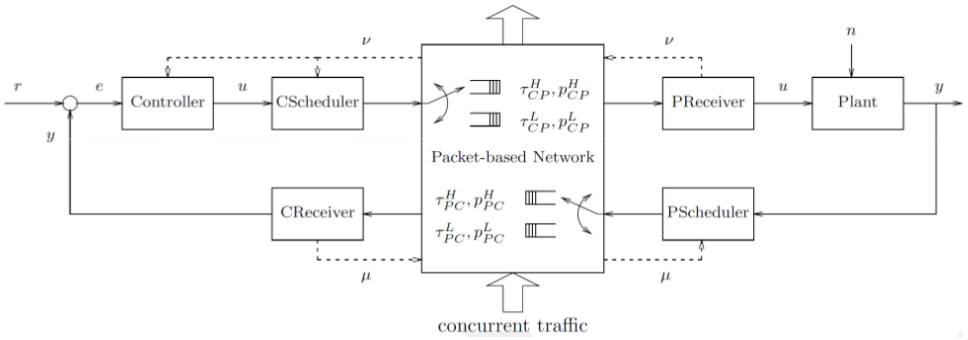


Figura 4.1: Architettura proposta di un sistema di controllo

tempo di campionamento Ts (il controllore), uno scheduler che decide la politica dei comandi per il controllore (CScheduler), uno scheduler che decide la politica delle misure per l'impianto (PScheduler), un ricevitore dalla parte del controllore che computa le stime su ritardo e PLR che devono essere mandate alla parte dell'impianto (CReceiver), un ricevitore dalla parte dell'impianto che computa le stime su ritardo e PLR che devono essere mandate alla parte del controllore (PReceiver).
I τ^H, τ^L sono le variazioni dei ritardi di trasmissione e i ρ^H , ρ^L sono il tasso di pacchetti persi.
I τ^H, τ^L sono valori di variazione temporale e la loro stima viene computata a run time confrontando il timestamp all'interno del payload e il tempo di arrivo (si assuma che i nodi siano sincronizzati).
I ρ^H, ρ^L sono valori di variazione temporale e la loro stima è ottenuta a run time contando i pacchetti arrivati e i pacchetti persi.
Vale la seguente relazione:

- $\tau^H \leq \tau^L$

- $\rho^H \leq \rho^L$

Si assuma che:

- i pacchetti mandati in ciascuna coda arrivano nell'ordine con cui sono stati mandati (questo vuol dire che un pacchetto puó essere superato solo da un pacchetto che appartiene ad una coda differente);
- I valori τ^H e τ^L e i valori ρ^H e ρ^L sono diversi nel passo controllore-impianto e il passo impianto-controllore;
- i valori di ritardo di τ^H e τ^L sono piú piccoli dell'intervallo di campionamento;

Gli scheduler scelgono la politica π_k per il k-pacchetto. Il CScheduler si occupa dei comandi dalla parte del controllore, il PScheduler si occupa delle misure dalla parte dell'impianto .
Per scegliere la politica π_k per il k-pacchetto l'algoritmo del CScheduler segue questi step:

1. Computa l'output stimato dell'impianto per la ricezione di successo (usando le politiche H e L) e per i pacchetti persi. Si tiene il comando precedente ogni volta che il comando corrente non arriva;
2. Calcola la differenza tra gli output stimati dell'impianto;
3. Confronta la differenza ottenuta con una soglia predefinita per scegliere la politica: π_k=H o π_k=L ;

La Network condition si ottiene pesando la stime sul comportamento dell'impianto attraverso le statistiche sui pacchetti persi, usate come probabilità *a-posteriori*:

$$\hat{\mathrm{y}}^H(s) = (1-\hat{\mathrm{p}}^H(k))\hat{\mathrm{y}}^H_{get}(s)+\hat{\mathrm{p}}^H(k)\hat{\mathrm{y}}_{lost}(s)$$
$$\hat{\mathrm{y}}^L(s) = (1-\hat{\mathrm{p}}^L(k))\hat{\mathrm{y}}^L_{get}(s)+\hat{\mathrm{p}}^L(k)\hat{\mathrm{y}}_{lost}(s)$$

La differenza tra gli output stimati é:

$$\hat{\mathrm{e}}(k) = \hat{\mathrm{y}}^L(s)-\hat{\mathrm{y}}^H(s)$$

Lo stato dell'impianto si ottiene osservando le misure e lo stato futuro stimato, registrando la strategia corrente di trasmissione:

if $(|\hat{\mathrm{e}}(k)| > E)$ {
  then $\pi_k = H$
  $\hat{\mathrm{x}}(k+1)|\pi_k) = \hat{\mathrm{x}}^H_{get}(k+1)$ }
else {
  $\pi_k = L$
  $\hat{\mathrm{x}}(k+1)|\pi_k) = \hat{\mathrm{x}}^L_{get}(k+1)$ }

Questa soluzione è vantaggiosa perchè i pacchetti appartenenti alla coda H sono usati quando bisogna mandare dati importanti oppure, nel caso in cui ci sia un canale molto occupato, si puó scegliere di mandare un pacchetto nella coda H nonostante questo non abbia una prioritá elevata. La classe ad alta priorità non viene sprecata perché, statisticamente, piú coppie controllore-impianto che condividono lo stesso canale non devono mandare pacchetti importanti nello stesso momento.

4.4 Soluzione 2: Progettazione di politiche di trasmissione in uno scenario di controllo di formazione

Dei veicoli autonomi dovrebbero adattare la loro traiettoria e la loro velocità per tenere le distanze l'uno dall'altro. Un leader sceglie la traiettoria. Bisogna tener conto inoltre dei vincoli spaziali e degli ostacoli. Ciascun veicolo riceve la posizione e la velocità del vicino attraverso messaggi wireless, calcola la sua posizione e velocità e cambia la traiettoria e la velocità a seguito di un comando. Ogni veicolo puó essere considerato un Network Control System (Figura 4.5):

Impianto: descrive il suo comportamento dinamico e cinematico;

Controllore e impianto sono direttamente connessi ;

Riferimento: la posizione del leader ricevuta attraverso la rete;

Pertubazioni: la sua posizione e velocità potrebbero cambiare per fattori esterni come vento, pioggia e ostacoli.

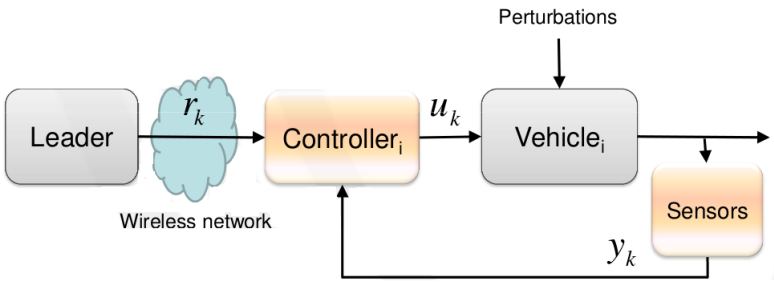


Figura 4.2: Controllo di formazione e NCS

Uno scenario semplificato, ma completo, prevede che ciascun veicolo:

- abbia un solo leader;

- mandi in broadcast la sua posizione e la sua velocità in modo tale che i veicoli successivi possano conoscerle;
- riceva tutti i pacchetti ma tiene solo quelli provenienti dal proprio leader;
- cambi la traiettoria e la velocità in base alla posizione e la velocità del proprio leader;

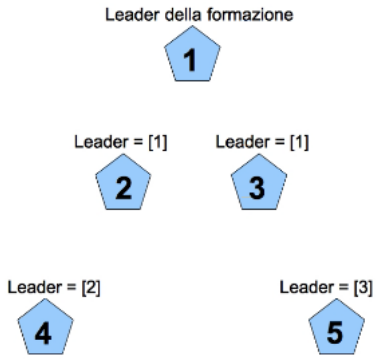


Figura 4.3: Scenario semplificato

Ci sono principalmente tre problemi ancora aperti:

1. I messaggi potrebbero non arrivare ai veicoli successivi, per collisioni tra i pacchetti oppure trasmissione out-of-range;
2. Se la posizione del leader non viene sentita i veicoli non possono reagire in tempo e collidere tra di loro, causando quindi la possibile perdita di veicoli;
3. Il tempo di reazione dipende dal momento in cui vengono ricevuti i pacchetti. è quindi influenzato da fattori come l'accesso al canale e il ritardo di propagazione, che è funzione della distanza.

Il controllore quindi deve considerare gli errori di posizione e di velocità e decidere la velocità, mentre la rete puó controllare la potenza di trasmissione, la politica di accesso al canale (TDMA vs CSMA), la priorità dei pacchetti e la ridondanza dei pacchetti. L'obbiettivo è quello di aggiustare la potenza di trasmissione per raggiungere i veicoli successivi evitando di disturbare gli altri e come strategia si puó usare l'informazione ricevuta dai veicoli successivi per stimare la loro distanza e aggiustare la potenza di trasmissione in base a questa. Questa stima quindi non è usata per il controllo della velocità, ma per cambiare la potenza di trasmissione in base alla *signal loss law*.

Signal loss law: Si assuma che la potenza del segnale decresca secondo alla seguente relazione $P_{RX} = \frac{1}{d^{\alpha}} P_{TX}$
Se $P_{RX} < P_{min}$ allora il pacchetto non viene ricevuto.

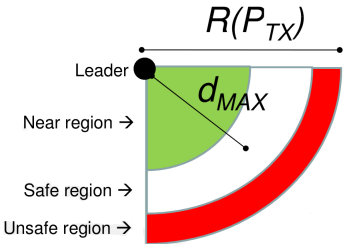


Figura 4.4: Range di trasmissione valido

Sia $R(P_{TX})$ il range di trasmissione valido in cui $P_{RX} > P_{min}$ mostrato in figura Figura 4.4.
Sia d_{MAX} la distanza massima tra i veicoli successivi.
Ad ogni istante di tempo:

- ascolta i messaggi relativi alle posizioni dei successivi;
- aggiusta P_{TX} in modo che d_{MAX} ricada nella regione $R(P_{TX})$

4.5 Simulazione per NCS

La simulazione della comunicazione deve prevedere:

- Matlab/Simulink, (Figura 4.5)il ritardo e la perdita dei pacchetti vengono iniettati in un collegamento punto-punto;
- Simulatore di rete, che descriva l'intera topologia e l'effetto con l'interazione con gli altri pacchetti.
- Collisioni
- Hidden node

I problemi per la co-simulazione sono:

Come connettere i diversi tool?
Possibili soluzioni: Socket o shared memory.

Come sincronizzare i thread della simulazione?
Possibili soluzioni: Eseguire un tool alla volta oppure prevedere un'esecuzione parallela con punti di sincronizzazione.

Come rappresentare le entità esterne in un modello?

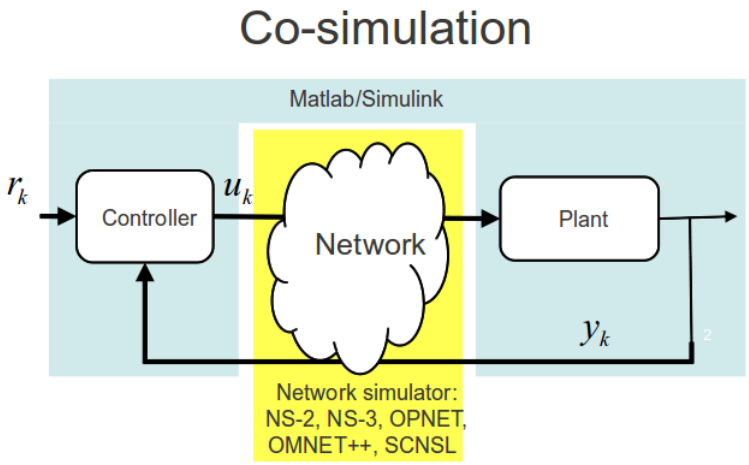


Figura 4.5: Co-simulazione

4.6 Relazione tra progettazione NCS e progettazione NES

In Figura 4.6 si vede che un Network Control System è organizzato come un anello chiuso.

Parlando invece di applicazioni distribuite di Networked Embedded Sy-

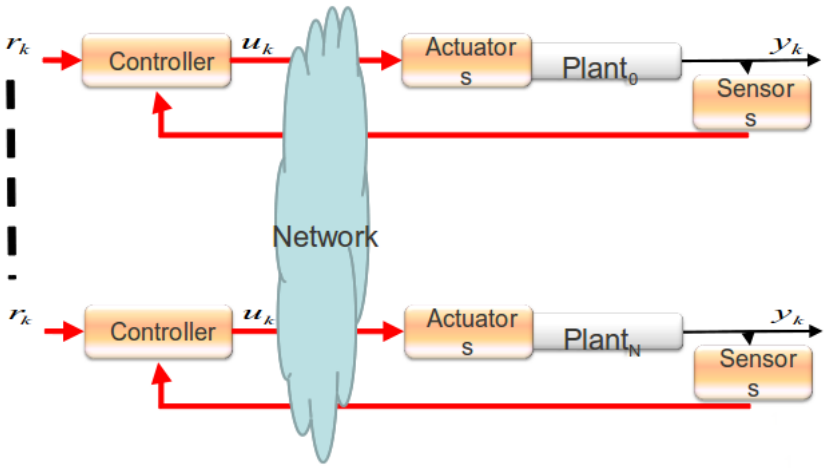


Figura 4.6: Insieme di networked control system

stem, come si vede in Figura 4.7, si ha a che fare con sistemi embedded che comunicano su una rete.

Si pone la questione perció su come mettere in relazione le due cose.

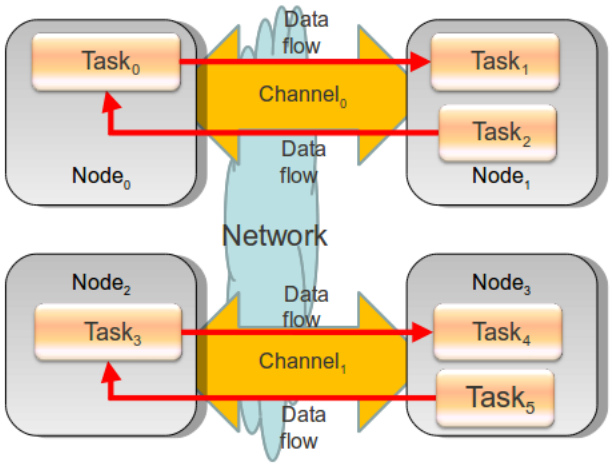


Figura 4.7: Applicazione distribuita di un networked embedded system

La risposta sta nel vedere le attivitá di sensing, attuazione e controllo della Figura 4.6 come specializzazioni dei task rappresentati in Figura 4.7 . Quindi si puó dire che uno o piú Networked Control System che utilizzano la stessa rete rappresentano un caso particolare di un'applicazione distribuita di un Networked Embedded System. Come conseguenza di questo ragionamento occorre precisare che le attivitá di sensing, attuazione e controllo, dopo essere state studiate dal punto di vista dei controlli automatici, devono essere realmente implementate su Networked Embedded System rispettandone vincoli di capacitá computazionale, memoria, consumo energetico, affidabilitá, e costo.

Discussione delle fonti

Le fonti bibliografiche di questa dispensa sono tratte in gran parte dai lucidi del corso di Sistemi Embedded di Rete dell'anno accademico 2011/2012.
Nella sezione 2.1 sono state raccolte alcune informazioni dal libro adottato durante il corso [1] e dalla tesi [2] .
Nella sezione 2.3 ci si riferisce invece allo standard [3] .
La parte sulle reti di campo descritta nella sezione 2.4 è tratta da diverse fonti: [4], [5], [6], [7] .
Il Capitolo 3 è stato sviluppato basandosi su [8] e, per quanto riguarda la parte di Network View Simulation, su [9] .
La sezione 4.3 viene presa da [10] .
La sezione 4.4 è realizzata grazie al contributo della tesi [11] .
La sezione 4.5, infine, è stata presa da [12] .

Bibliografia

[1] Andrew S. Tanenbaum, Reti di calcolatori (Edizione 4) Pearson - Prentice Hall 2003.

[2] Marco Beltrame, *Sviluppo di una rete wireless di sensori per il monitoraggio di strutture in tempo reale*, netlab-mn.unipv.it/thesis/MSc/Beltrame_Thesis.pdf

[3] *LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information technology -Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personl Area Networks (LR WPANs)* September 2006

[4] *Application Layer and Communication Profile CiA Draft Standard 301* Version 4.02

[5] H. Boterenbrood, *CANopen high-level protocol for CAN-bus*

[6] Daniele Gallinella, *CAN Controller Area Network*

[7] Giuseppe Tognini, *Studio, progetto e realizzazione di un analizzatore di traffico su rete CAN*, www.auto-consulting.it/DWN/Tesi_CAN_Analyzer.pdf

[8] E. Ebeid, F. Fummi, D. Quaglia, F. Stefanni, *Refinement of UML MARTE models for the design of networked embedded systems, Atti di "Design, Automation and Test in Europe Conference and Exhibition (DATE)"*, Dresden, Germany, March 12-16, 2012, EDAA, pp. 1072-1077

[9] D. Quaglia, F. Stefanni, F. Fummi, *SystemC Network Simulation Library*, http://sourceforge.net/projects/scnsl/

[10] D. Quaglia, R. Muradore, P. Fiorini, *Plant control over QoS-enabled packet networks, IEEE International Symposium on Industrial Embedded Systems (SIES)*, pp. 132-139, Vasteras, Sweden, 15-17 June 2011.

[11] Progettazione di politiche di trasmissione in uno scenario di controllo di formazione, E. Marcolongo, G. Lanza, D. Quaglia, R. Muradore, *Progettazione della politica di trasmissione wireless in un problema di controllo di formazione mediante co-simulazione*, Dipartimento di Informatica, http://www.di.univr.it/report

[12] D. Quaglia, R. Muradore, R. Bragantini, P. Fiorini, *A SystemC/-Matlab co-simulation tool for networked control systems. Simulation Modeling Practice and Theory, Elsevier* , April 2012, vol. 23, p. 71-86

Elenco delle figure

University of Verona
Department of Computer Science
Strada Le Grazie, 15
I-37134 Verona
Italy

http://www.di.univr.it