TinyOS

Sistema operativo open-source per wireless sensor network sviluppato dall'Università di Berkley e centro ricerche Intel
www.tinyos.net

Enrico Oliva

Outline

Outline

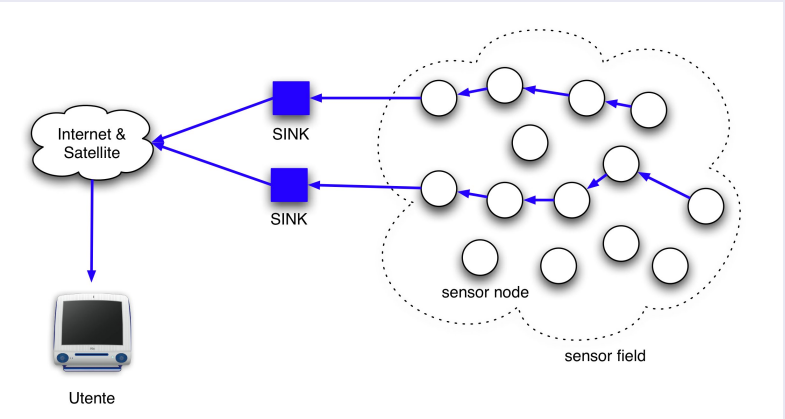
Wireless Sensor Network

WSN definition

sistema di devices - dotate di processore, capacità comunicative e percettive - organizzato in una rete wireless.

Caratteristiche e criticità di una WSN

- Budget energetico limitato
- Tolleranza ai guasti
- Scalabilità
- Topologia variabile
- Elaborazione in tempo reale
- Costi di produzione bassi
- Problemi di sicurezza
- Scarse risorse hardware



Un sistema operativo per le reti di sensori

Requisiti funzionali di un Sistema Operativo per WSN

- Supporto efficiente alla concorrenza (per ottimizzare l'uso delle limitate risorse e vincoli real-time)
- Flessibilità e modularità (per adeguarsi alle diverse piattaforme e scenari applicativi)
- Uso efficiente delle risorse (per ottimizzare il consumo energetico)

Due tipologie di Sistema Operativo per WSN

- Sistemi operativi con architettura *Component-Based*: *TinyOS* e *Btnode*
- Sistemi operativi con architettura *General-purpose* ridotti: *MANTIS* e *BerthaOS*

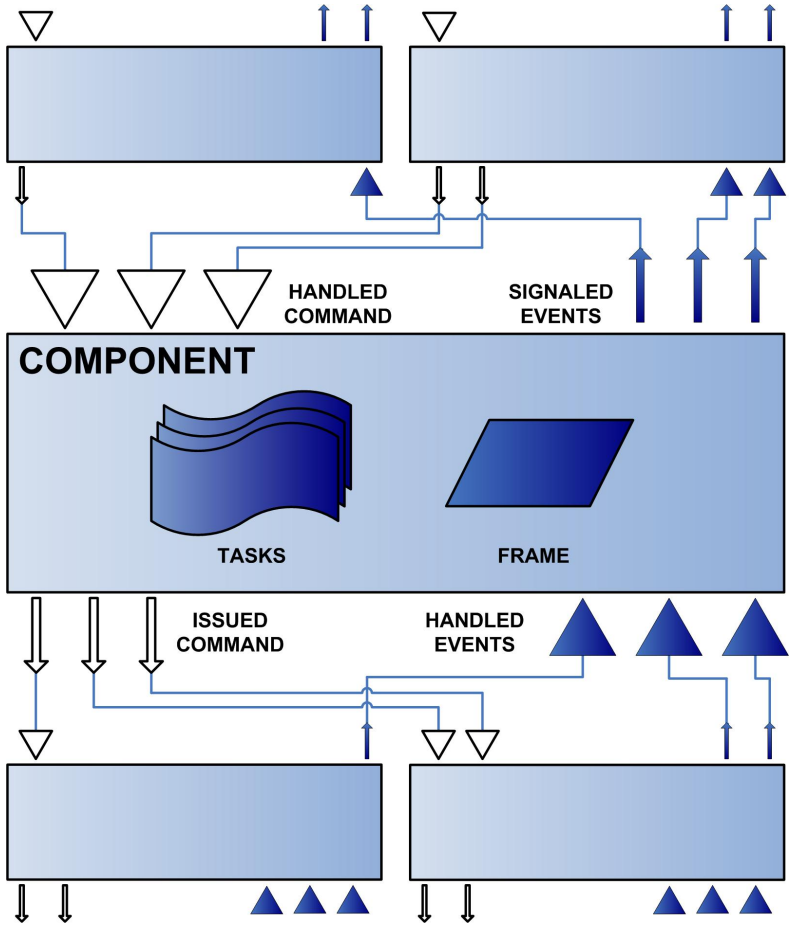
Outline

TinyOS è un “piccolo” sistema operativo

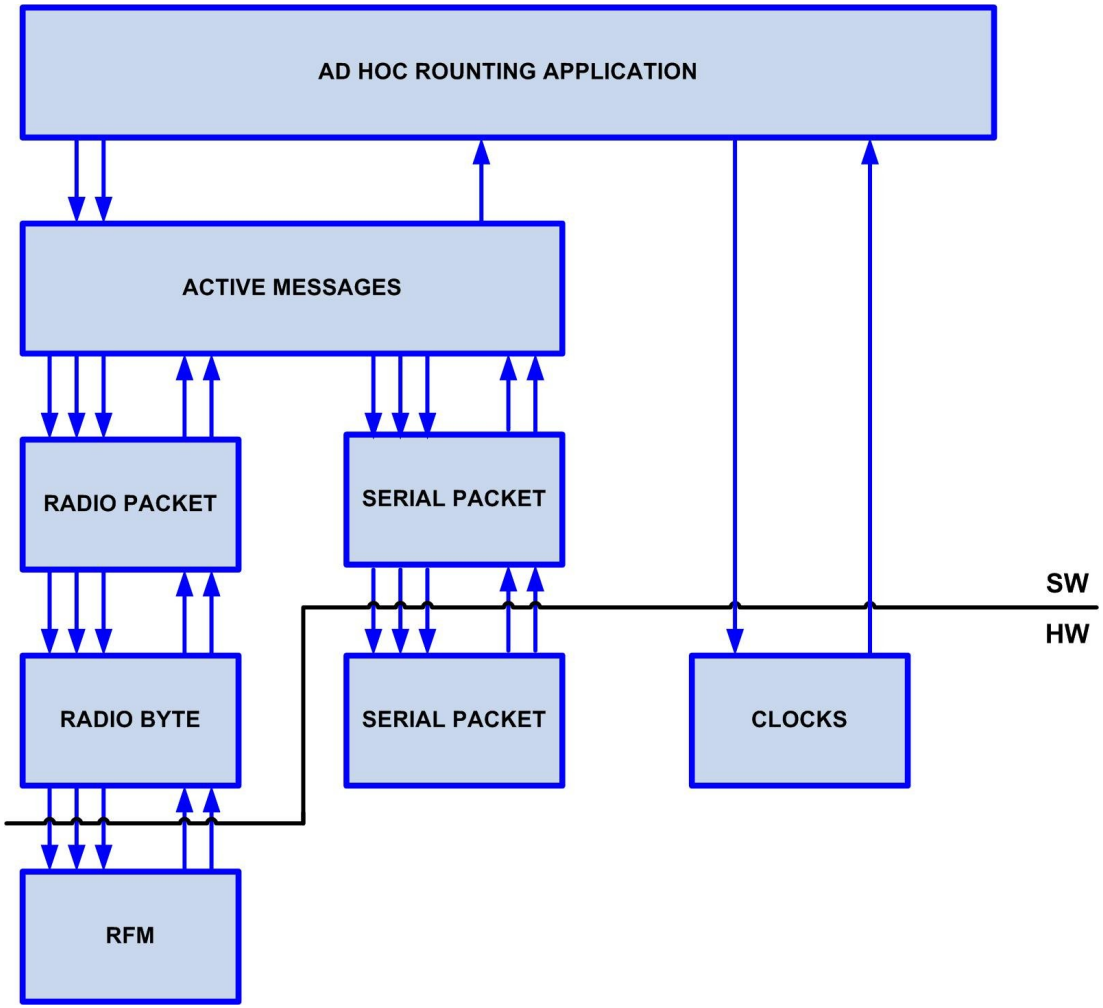
- Dimensioni fortemente ridotte: una tipica applicazione TinyOS-Based che occupa una decina di KB solo 400B (circa) competono al sistema operativo
- Sistema operativo o *programming framework*?
  - Libreria di componenti legacy: protocolli di rete, servizi distribuiti, driver sensor, tool acquisizione dati
  - Versione **application-specific** di TinyOS a seconda delle necessità
  - Esporta allo strato applicativo un’astrazione dello strato hardware - gestendo risore fisiche e software
- Architettura *component-based* e programmazione *event-driven*
- Linguaggio NesC - simile al C per applicazioni concorrenti

Componente in TinyOS

- Il modello di componente segue quello generale della Component Based Software Engineering
- Ogni componente è specificato medinate le interfacce che importa e esporta
- Elementi costitutivi
  - Frame
  - Comandi
  - Event Handler
  - Task



Applicazione = Grafo dei componenti



Divide et Impera!

Modello di esecuzione

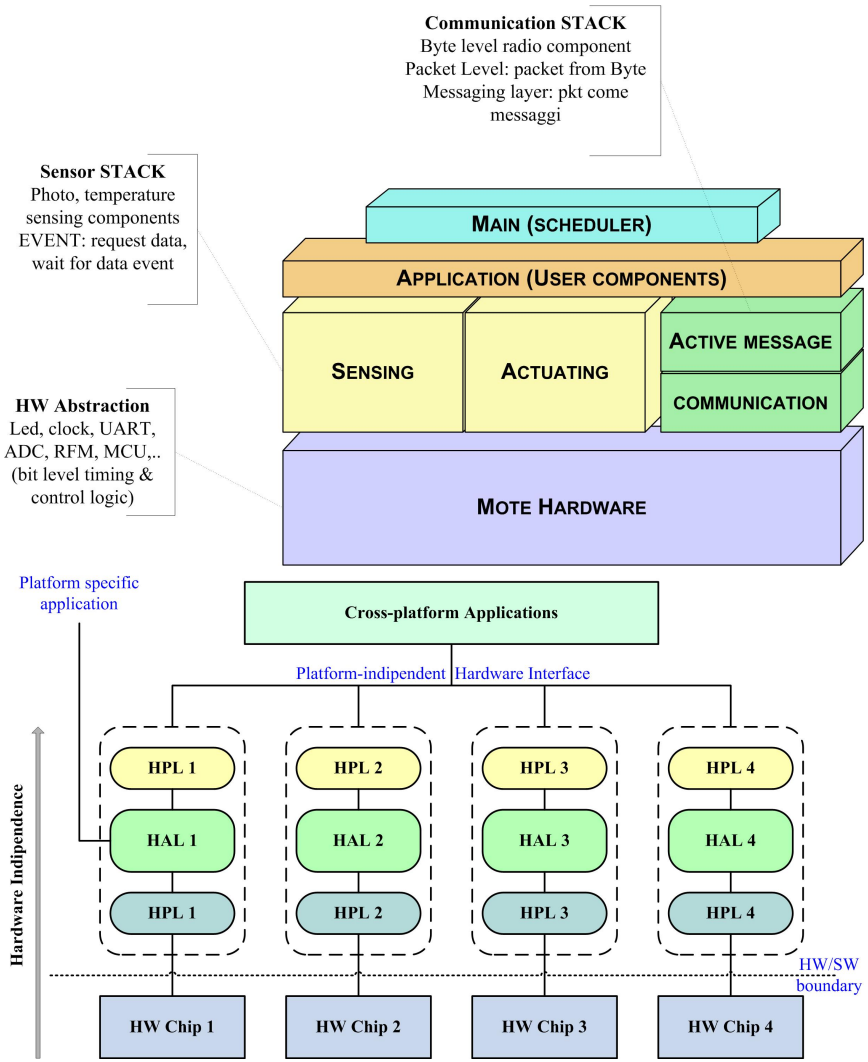
- Modello concorrente event-based
  - Si integra alla natura fortemente reattiva dei nodi sensori
  - Si ha una riduzione di un fattore 2 e 30 della memoria per istruzioni e dati *[Low Power Operating System for Heterogeneous Wireless Communication Systems]*
  - Si riducono i consumi energetici stimati in un fattore 12
  - Modello process-based inadeguato -> *context-switch*
- TinyOS 2.0 usa un modello event-based con scheduling a due livelli gerarchici e due differenti contesti
  - per la gestione degli eventi time-critical -> eventi hanno diritto di prelazione reciproco e sui task
  - per il processamento del normale codice -> task hanno un unico livello di priorità run-to-completion

Split-phase operation

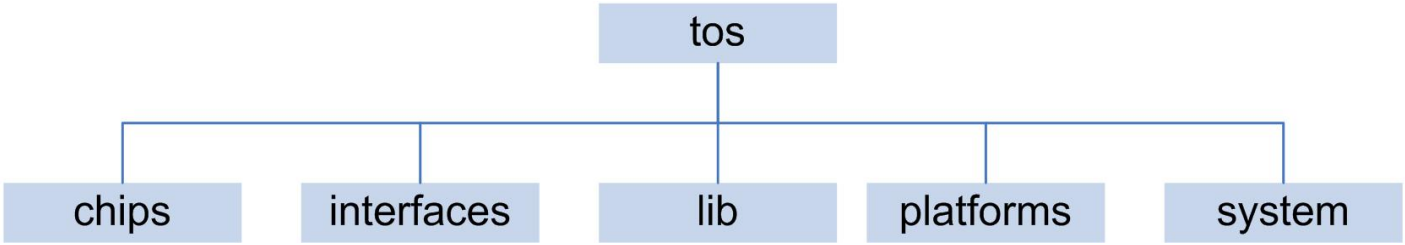
Separazione tra invocazione di un servizio e operazione di ritorno: comando-task-evento

Architettura

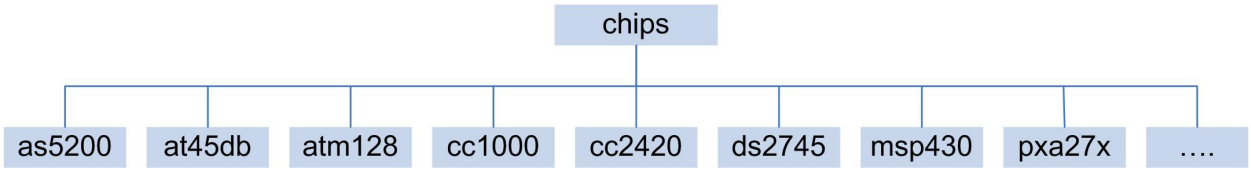
- Decomposizione in layer verticali e orizzontali
- Hardware Abstraction Architecture (per la portabilità)
  - Hardware Presentation Layer
  - Hardware Adaptation Layer
  - Hardware Interface Layer
- Utilizzo dei pattern OO: *Decorator*, *Service Instance*, *Facade* e *Placeholder*



Struttura delle directory



- tos/chips - codice dei chip utilizzati



- tos/platform - codice delle piattaforme
- tos/systems - codice dei componenti di sistema
- tos/interfaces - codice delle interfacce core HIL
- tos/lib - codice di libreria

Piattaforme supportate dal TinyOS

Node	CPU	Memory	Communication
BTnode 3	Atmel ATmega 128L 8 bit RISC 8 MHz	4KB RAM 128 KB Flash ROM	Chipcon CC1000 ISM band 433-915 MHz 38.4 Kbit/s Bluetooth Zeevo ZV4002
Imote	ARM7TDMI3 32 bit RISC 12 MHz	64KB RAM 512KB FLASH ROM	Zeevo Bluetooth TC2001P
Imote2	Intel PXA271 13-416MHz MMX DSP Coprocessor	256KB RAM 32MB FLASH ROM	Chipcom CC2420 IEEE 802.15.4 250 Kbit/s
Mica	Atmel ATmega 128L 8 bit RIS 8 MHz	4KB RAM 128 KB FLASH ROM	RF Monolitich TR1000 ISM band 433-915 MHz 30 to 115.2 Kbit/s
Mica2	Atmel ATmega 128L 8 bit RIS 8 MHz	4KB RAM 128 KB FLASH ROM	Chipcon CC1000 ISM band 433-915 MHz 38.4 Kbit/s
Mica2Dot	Atmel ATmega 128L 8 bit RISC 8 MHz	4KB RAM 128 KB FLASH ROM	Chipcon CC1000 ISM band 433-915 MHz 38.4 Kbit/s
MicaZ	Atmel ATmega 128L 8 bit RISC 8 MHz	4KB RAM 128 KB FLASH ROM	Chipcom CC2420 IEEE 802.15.4 2.4GHz Band 250 Kbit/s
Telos	Texas Instruments MSP430F1611 16 bit RISC 4-8Mhz	10KB RAM 48 KB FLASH ROM	Chipcom CC2420 IEEE 802.15.4 2.4GHz Band 250 Kbit/s
	Texas Instruments		Infineon TDA 5250

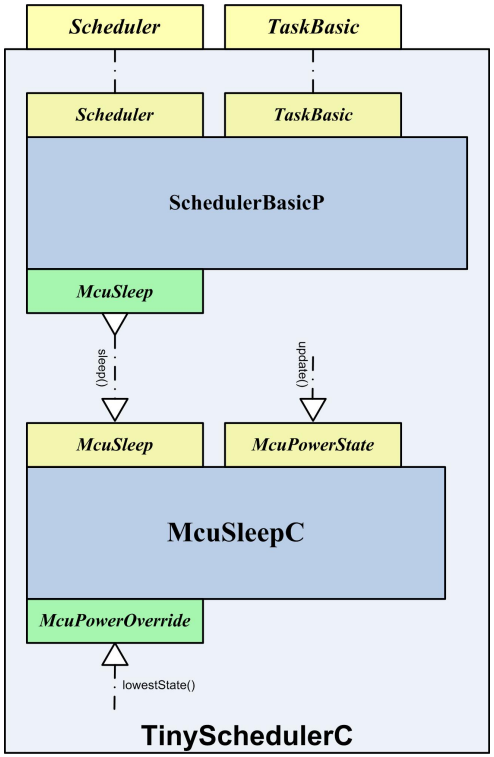
Linguaggio NesC: overview

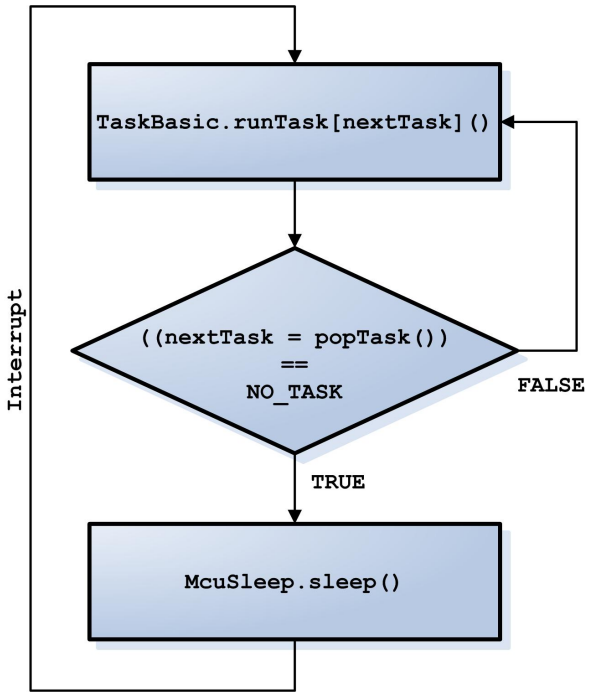
Network Embedded System C (NesC)

è un “linguaggio statico” che non prevede nessun meccanismo di allocazione dinamica della memoria

- Codice è “rotto” in unità funzionali: i componenti (configurazione/moduli)
- Namespace locale - visibilità a livello di componente
- Ottimizzazioni a compile-time
  - eliminazine del codice morto
- Ogni interfaccia è bidirezionale: comandi/eventi
- Separazione della costruzione dalla composizione
  - componenti assemblati staticamente
- Modello di concorrenza di NesC aderisce a quello del TinyOS

Dynamic Power Management (DPM) in TinyOS





- Componenti per la gestione energetica sono: Scheduler e McuSleep
- Comando McuSleep.sleep() porta il microcontrollore al più basso stato energetico compatibile con i requisiti applicativi
- Esperimento: corrente erogata dalle batterie 22.4 mA in trasmissione - 22 μA dopo l'esecuzione comando di sleep

Thank you!

Questions?