Sistemi Operativi per Sistemi di Elaborazione Ubiqui

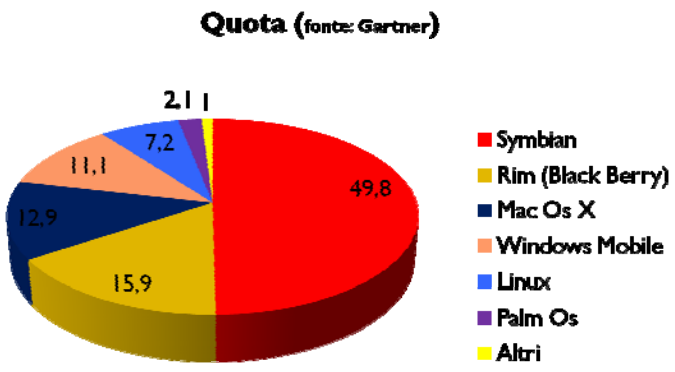
Sistemi Operativi per Ubiquitous Computing

- Palm OS
- Symbian OS
- Windows Mobile
- Embedded Linux
- QNX Neutrino
- TinyOS
- Android

Sistemi Operativi per Ubiquitous Computing

- I sistemi operativi per sistemi di elaborazione ubiqui seguono i principi dei SO classici ma
  - devono gestire risorse con caratteristiche particolari,
  - hanno interfacce molto diverse
- In questo settore non esiste un SO prevalente. I più diffusi sono Symbian, Rim, Mac OS, Palm OS e Windows Mobile insieme a TinyOs e Embedded Linux.

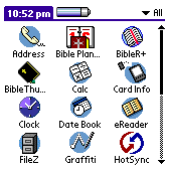
Quote di Mercato dei Sistemi Operativi Mobili



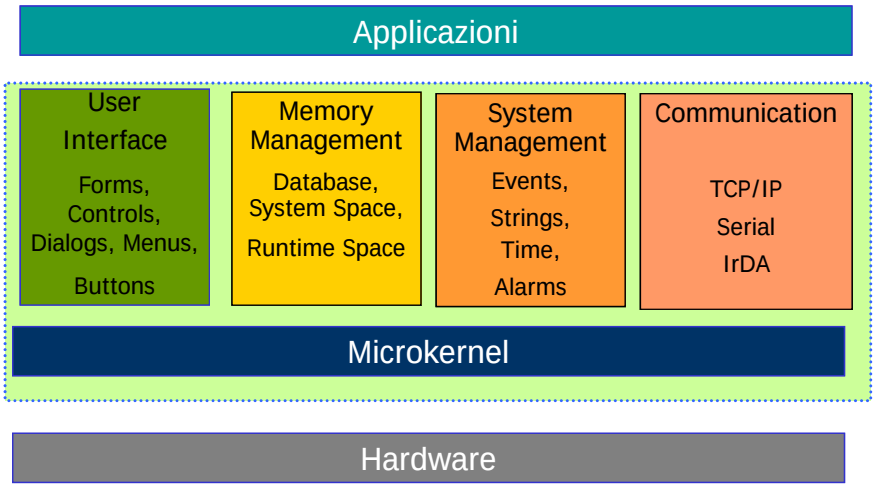
Sistemi Operativi : Palm OS

- Sistema operativo per cellulari e palmari (PDA).
- Sistema operativo a 32 bit

- Versioni correnti : (Palm OS Garnet 5.x, Palm OS Cobalt, version 6.1) con supporto Bluetooth e 65K colori per PDA/cellulari multimediali.



Sistemi Operativi : Palm OS



Sistemi Operativi : Palm OS

- **User Interface**: gestione dell'I/O grafico, menu, buttons, forms.
- **Memory Management**: DB, runtime space, system space, variabili globali.
- **System Management**: eventi, alarms, time, strings, …
- **Communication Layer**: I/O seriale, TCP/IP, Infrared Data Association (IrDA).

Sistemi Operativi : Palm OS

- **User management**: SO single user.
- **Dimensione** : v3.5 richiede circa 1.4 MBytes.
  La versione Cobalt:
  - Per cellulari richiede 32MB SDRAM + 32MB Flash
  - Per PDA 32MB SDRAM + 16MB ROM
- **Task Management**: multitasking e multithreading.
- **Power Management**: tre stati (sleep, doze, running)

Sistemi Operativi : Palm OS

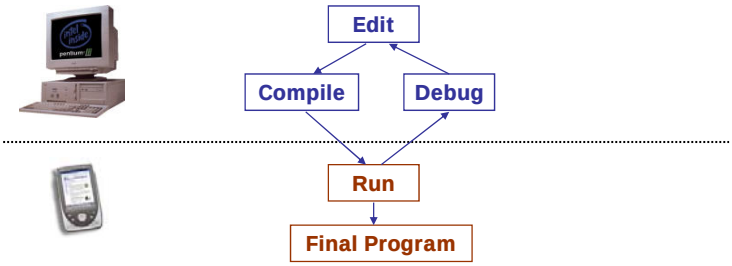
- **Memory Management**: le applicazioni non sono separate (una applicazione può causare il crash del sistema). Il file system tradizionale è sostituito da un insieme di database gestiti da un *Database Manager*
- La memoria è separata in
  - *Dynamic heap* : dimensione tra 64Kb e 256Kb e serve a contenere le variabili globali, lo stack, e la memoria allocata dinamicamente durante l'uso.
  - *Storage* : contiene dati permanenti (DB, files) che non vanno cancellati allo spegnimento.

Sistemi Operativi: Palm OS

- Palm OS è un sistema multi-task *event driven*.
- Gli eventi possono essere:
  - Una azione della user interface (es., touch screen op.)
  - Un system notification (es., un alarm del timer)
  - Un evento di una applicazione (es., richiesta di search)

Sistemi Operativi: Palm OS – Sviluppo SW

- Sviluppo del Software in C e C++ con CDK e SDK esterno.



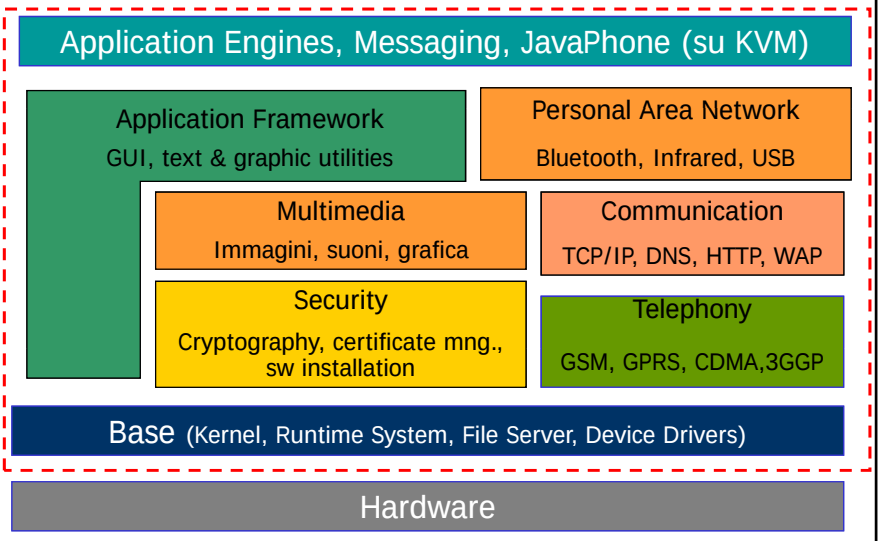
- Esiste un Palm Emulator per sviluppare e fare il test delle applicazioni prima di eseguirle su Palm OS
- Macchine virtuali: KVM, J9, WabaVM (poco efficienti)

Sistemi Operativi: Symbian OS

symbian

- Creato con il nome di EPOC da Psion come Sistema Operativo per telefonia mobile.
- Attualmente sviluppato da Symbian.
- Usato in cellulari NOKIA e Sony Ericsson e su diversi processori (anche in emulazione).
- Caratteristiche: multi-tasking real-time pre-emptive, 32 bit.

Sistemi Operativi : Symbian OS

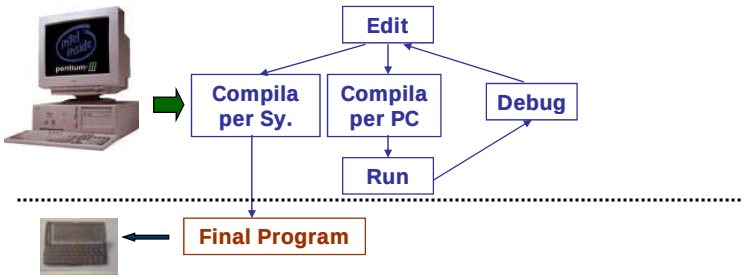


Sistemi Operativi : Symbian OS

- **User management**: SO single user.
- **Task Management**: microkernel real-time, multitasking con scheduling pre-emptive e con priorità. (Gestione complessa di applicazioni)
- **User Management**: con interfaccia standard: grafica, suoni e tastiera.
- **Memory Management**: MMU con spazi di indirizzi separati per applicazioni.

Sistemi Operativi: Symbian OS

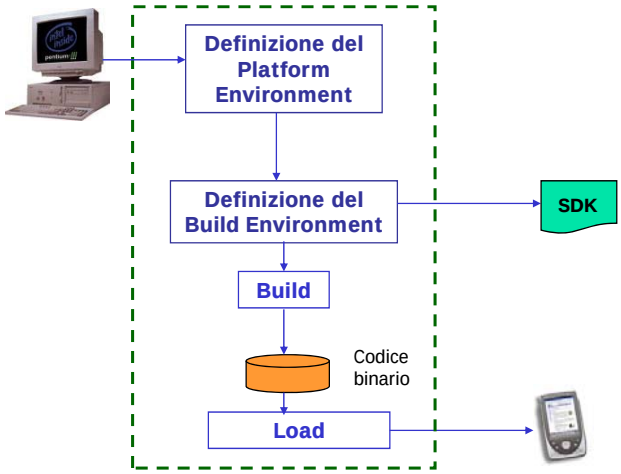
- Sviluppo del Software in C++, Java e OPL (Basic-like).
- Esiste un Simulatore per sviluppare e fare il test delle applicazioni prima di eseguirle su Symbian OS



Sistemi Operativi: Windows Mobile

- Windows Mobile è una versione di Windows sviluppata per sistemi mobili e pervasivi.
- Il sistema va configurato per la specifica piattaforma (PDA, cellulare, altro) su cui deve essere usato. Basato su memoria ROM.
- Ha l'interfaccia tipica di Windows adattata per i display dei sistemi mobili.

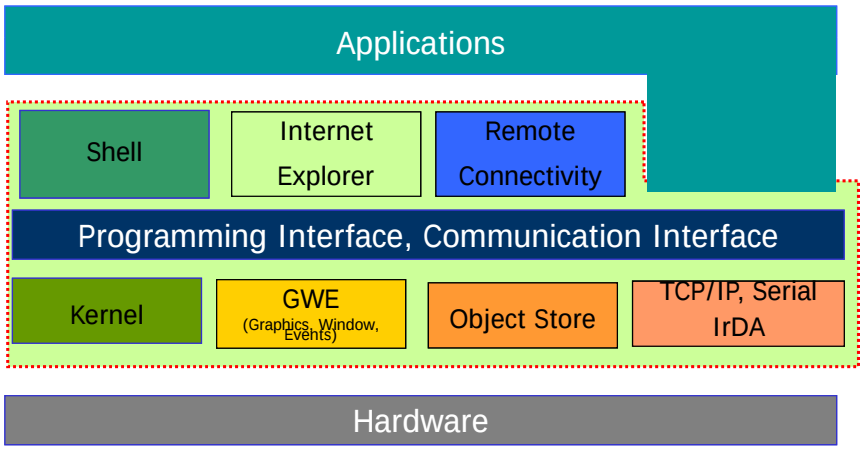
Sistemi Operativi: Windows Mobile – Configurazione



Sistemi Operativi: Windows Mobile

- **User management**: SO single user.
- **Task Management**: fino a 32 processi e un numero elevato di thread (limitato dalla memoria disponibile)
- **User Interface**: icone, dialog boxes, menu, suoni (approccio alla Windows)
- **Memory Management**: memoria protetta con 32 MB per processo, heap per file system, registry e object store (fino a 256 MB).

Sistemi Operativi : Windows Mobile



Sistemi Operativi: Windows Mobile

- **Dimensione** : La versione CE occupava da 400 Kb (kernel) a 3 MB (sistema completo) fino a 8 MB con Pocket Word e Internet Explorer.
- **Security**: Crittografia con una libreria per gestire i dati memorizzati in sicurezza. Memorizzazione sicura con smart card.
- **Ambienti Software** : Visual C++, Visual Basic, KVM, J9 e Waba.

Sistemi Operativi: Embedded Linux

- Versione di Linux per sistemi di elaborazione ubiqui.
- Architettura a microkernel. Funzioni e servizi compilabili nel kernel o generabili come moduli separati da caricare dinamicamente.
- Ampia gamma di protocolli e servizi per il networking.
- Configurabile e scalabile da un orologio ad un multiprocessore.

Sistemi Operativi: Embedded Linux

- **User management**: SO multi user.
- **Task Management**: Multitasking con scheduler pre-emptive e real-time (opzionale). Supporto per multiprocessori.
- **User Interface**: basata su X-Window
- **Memory Management**: gestione alla Linux con MMU e memoria virtuale.
- **Dimensione** : da 200 Kb (kernel) a circa 10 MB.

Sistemi Operativi: Embedded Linux

- La specifica è basata su:
  - Linux Standards Base 1.2.
  - IEEE POSIX 1003.1-2001 specification, contenente funzionalità Realtime, Threads and Networking.
  - Single UNIX Specification v3.

Sistemi Operativi: Embedded Linux

- L'ambiente di sviluppo per Linux è disponibile in Embedded Linux.
- Linguaggi: C, C++, Java
- Driver, utility, protocolli e programmi client e server disponibili per connessioni Internet.
- Similitudini con QNX Neutrino: sistema operativo POSIX compliant.

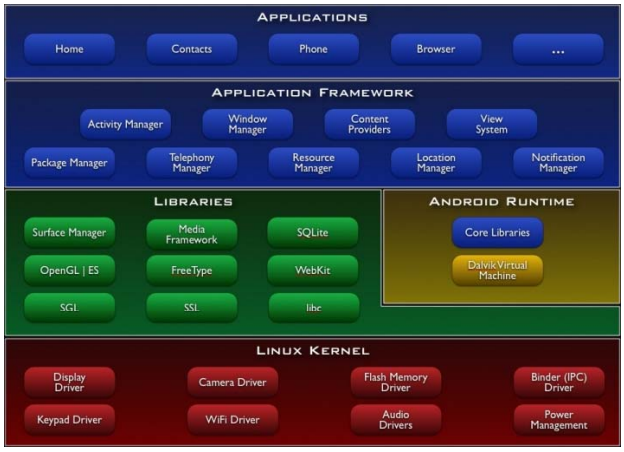
Sistemi Operativi: Embedded Linux on a watch

- Embedded Linux in un orologio
- Comunicazione wireless con altri dispositivi, PIM (calendario, address book, event list, …).
- Connessione ad Internet.
- Processore ARM7, 8MB memoria flash, 8MB memoria DRAM, infrarossi, RF wireless, 96x112 touch screen, roller wheel.

Sistemi Operativi e altro: Android

Android è un ambiente software (Sistema Operativo, middleware, applicazioni) per device mobili sviluppato da Google e basato su Java.

code.google.com/android



Android: Caratteristiche

- **Application framework** per il riuso e la gestione delle componenti
- **Dalvik virtual machine** ottimizzata per mobile devices
- **Integrated browser** basata sulla open source WebKit engine
- **Optimized graphics** con 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification
- **SQLite** per il data storage
- **Media support** per audio, video, formati (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony**
- **Bluetooth, EDGE, 3G, and WiFi**
- **Camera, GPS, accelerometro**
- **Rich development environment** che include un device emulator, tool di debugging, memory e performance profiling, and un plugin per Eclipse.

Android: Open Source

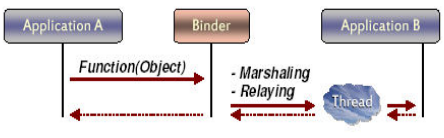
- La maggior parte dei componenti di Android sono distribuiti sotto la licenza Apache 2.0, ma ci sono alcune parti, come ad esempio il Kernel Linux, sotto la licenza GPLv2.
- Essendo Android open source, può essere esteso per includere nuove tecnologie e continuare ad evolvere grazie alle applicazioni create dalla comunità di sviluppatori.
- Android non fa distinzione tra gli elementi base di un cellulare e applicazioni di terze parti. Tutti possono accedere in modo uguale alle risorse del telefono, integrarsi tra di loro, scambiandosi dati e attivandosi a vicenda, e l'utente può decidere di sostituire le applicazioni originali. (RISCHIO!!!)

Android: Linux Kernel

- Android si appoggia sulla versione 2.6 del Kernel di Linux per servizi del sistema centrale, come sicurezza, gestione della memoria, gestione dell'hardware sottostante, compresi i driver per connettività (GSM, Bluetooth e WiFi), le interfacce di I/O, i dispositivi per il GPS, videocamera, ecc.
- Tuttavia non è esattamente il Kernel Linux: non c'è il supporto della libreria glibc e non è presente il nativo "windowing system".
- Rispetto al Kernel Linux sono state effettuate altre aggiunte: Alarm, Android Shared Memory, Kernel Memory Killer, Kernel Debugger, Logger, Power management e Binder (driver per interprocess communication veloce)

Android: Il Binder

- Il Binder traccia anche i riferimenti degli oggetti condivisi tra i processi in modo da garantire che non vengano distrutti finché qualche processo ha un riferimento ad essi.
- La comunicazione è sincrona, infatti quando un processo A effettua una chiamata a B, A si blocca finché non riceve una risposta.



- Per la comunicazione tra processi che sviluppa può utilizzare l'AIDL (Android Interface Definition Language) un IDL usato per generare un codice che permette a due processi su un dispositivo Android di comunicare usando IPC.

Android: Libraries

- Consiste di un set di librerie C/C++ tra cui abbiamo Bionic (200 K), che è la versione modificata da Google della libreria libc.
- Sono incluse una serie di librerie funzionali. Google ha scelto come browser Web l'open source Webkit.
- Android Media Framework è una libreria basata sulla piattaforma PacketVideo OpenCORE e supporta i più importanti standard audio e video. Utilizza inoltre delle librerie per grafica 2D e 3D basate su openGL ES permettendo così di creare applicazioni come Google Earth.
- Tra librerie funzionali è presente anche SQLite, un motore di database relazionale potente e leggero, disponibile per tutte le applicazioni.

Android: Application Framework

- A questo livello gli sviluppatori hanno pieno accesso alle stesse framework API usate dalle applicazioni di base.
- L'architettura delle applicazioni è progettata per semplificare il riutilizzo dei componenti; ogni applicazione può rendere pubbliche le sue capacità e tutte le altre possono quindi farne uso, con i limiti imposti dalla sicurezza del framework.
- Alla base di ogni applicazione si trova un set di servizi e sistemi, tra cui:
  - **un estendibile insieme di Viste, che possono essere usate per costruire un'applicazione, come liste, caselle di testo, pulsanti, ecc.;**
  - **dei Content Providers, che permettono alle applicazioni di accedere ai dati di altre applicazioni (come i Contatti), o di condividere i propri;**
  - **un Manager delle risorse, che offre l'accesso a risorse non-code come stringhe localizzate, grafica, files di layout, file audio;**
  - **un Manager delle notifiche, che permette a tutte le applicazioni di mostrare avvisi personalizzati nella status bar;**
  - **un Manager delle attività, che gestisce il ciclo di vita delle applicazioni e fornisce un backstack di navigazione comune;**
  - **un Package Manager, che mantiene le informazioni sulle applicazioni caricate nel sistema.**
  - **Ci sono inoltre gli Hardware Service che forniscono l'accesso alle API di basso livello dell'hardware**

Android: Applications

- Il quarto livello è quello delle Applicazioni.
- Android funziona con un set di applicazioni di base che comprende un email client, un programma sms, calendario, mappe, browser, contatti e altro. Tutte le applicazioni sono scritte in linguaggio Java e sono espandibili.

Android: Applications

- Ci sono quattro blocchi fondamentali che costituiscono un'applicazione Android:
  1. Activity (singola schermata che presenta un'interfaccia utente composta da Views e risponde agli eventi)
  2. Broadcast Intent Receiver (usato quando si vuole che un'applicazione venga eseguita in reazione ad un evento esterno)
  3. Service (codice long life che gira senza un'interfaccia utente. Un esempio è un Player che esegue delle canzoni da una lista)
  4. Content Provider (permette che i dati dell'applicazione vengano condivisi con altre ed implementa un set standard di metodi che permettono ad altre applicazioni di immagazzinare e recuperare i dati)
- Non tutte le applicazioni hanno bisogno di tutti i blocchi, ma sono sicuramente formate da una loro combinazione.
- Ogni applicazione ha un file XML, AndroidManifest.xml in cui sono elencati i componenti usati, le loro funzionalità e i loro requisiti.

Android: Processi

- Nella gran parte dei casi, ogni applicazione Android gira sul suo proprio processo Linux. Questo è creato quando si deve eseguire parte del codice dell'applicazione e continuerà ad essere in esecuzione fino a quando non sarà più necessario.
- Per determinare quali processi devono essere portati a termine quando il sistema è a corto di memoria, Android li colloca in una "gerarchia di importanza" in base ai componenti in esecuzione e ai loro stati.
- I tipi di processi sono:
  - **foreground process**
  - **visible process**
  - **background process**
  - **empty process**

Android: La Dalvik Virtual Machine

- La Dalvik Virtual Machine è stata progettata considerando i limiti della piattaforma ed in particolare una CPU non molto potente e non molta RAM.
- La Dalvik virtual machine esegue applicazioni Java che sono state convertite in un formato compatto, il Dalvik Executable (.dex), adatto per sistemi che sono limitati nella memoria e nella velocità del processore.
- Android ha bisogno di almeno 64 MB di RAM; all'avvio del sistema, dopo aver caricati tutti i servizi di alto livello, vengono occupati 40MB alle applicazioni restano circa 20MB.

Android: La Dalvik Virtual Machine

- A differenza della Java Virtual Machine, la Dalvik è basata su registri e questo permette di avere un interprete molto più efficiente in quanto ogni istruzione interpretata è semanticamente più "densa".
- Un altro vantaggio di utilizzare la macchina basata su registri si ha quando si deve accedere più volte ad una stessa variabile dentro un ciclo.
- In una macchina basata su stack la variabile viene messa nello stack ad ogni iterazione, con una macchina a registri una volta caricata la variabile nel registro, questa può essere riusata più volte fino alla fine del metodo.

Android: Programmazione

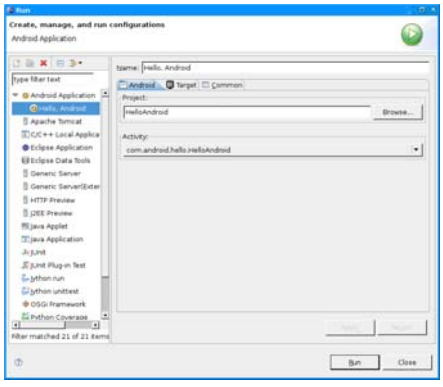
- Per programmare una applicazione in Android occorre creare un progetto tramite Eclipse:
  1. Creaee un "Android Project" tramite il menu **File > New > Project**
  2. Inserire I dettagli nel New Android Project dialog.
  3. Editare il template auto-generato di codice sorgente secondo quello che si vuole ottenere.

Android: Programmazione

```
public class HelloAndroid extends Activity {
       /** Called when the activity is first created. */
       @Override
       public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          }
}
```

```
package com.android.hello;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class HelloAndroid extends Activity {
          /** Called when the activity is first created. */
          @Override
          public void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    TextView tv = new TextView(this);
                    tv.setText("Hello, Android");
                    setContentView(tv);
          }
}
```


Android: Programmazione



Sistemi Operativi per Sensori

- Un sistema operativo per sensori deve possedere alcune caratteristiche base:
  - dimensioni ridotte,
  - basso consumo durante l'elaborazione,
  - consumo quasi nullo durante lo stato di *idle*,
  - gestione della concorrenza,
  - implementazione di protocolli di rete a seconda della periferica di rete utilizzata
  - fornire un astrazione per i dispositivi hardware (sensori ed attuatori), montati sul nodo sensore.
- Ad esempio un protocollo di trasporto come il TCP/IP non può essere facilmente impiegato in quanto offre un servizio orientato alla connessione , e siccome un nodo sensore possiede limitate capacità , anche di memoria , un protocollo di questo tipo è attualmente poco applicabile.

Sistemi Operativi per Sensori

Esistono 2 approcci allo sviluppo di sistemi operativi per sensori:

- Sviluppare un sistema i cui componenti vengono compilati insieme all'applicazione.
  - Questo consente che ci sia una singola applicazione in esecuzione in un dato momento con riduzione dei consumi energetici per *l'assenza di context-switch* e sistemi molto piccoli (nel sistema sono caricati solo i moduli che effettivamente verranno utilizzati).
  - Lo svantaggio principale riguarda la versatilità ed i vincoli di riconfigurabilità dell'applicazione.
- Sviluppare un sistema che includa i tradizionali strati di software di un sistema operativo tradizionale in versione ridotta.
  - In questo caso è difficile tenere sotto controllo i consumi e le risorse impiegate, ma si guadagna in termini di versatilità visto che ci possono essere, più applicazioni in "esecuzione".

Sistemi Operativi per Sensori: TinyOS

- TinyOS consiste in un insieme ridotto di moduli (servizi) software che forniscono funzionalità per il controllo di un nodo sensore.
- Quando un'applicazione viene compilata, i componenti di TinyOS vengono compilati insieme ad essa ed il risultato costituisce l'intero software del sensore, consentendo così un risparmio di energia e di memoria.
- Non è possibile installare più applicazioni indipendenti sullo stesso sensore.

Sistemi Operativi per Sensori: TinyOS

- In TinyOS non esiste un kernel che gestisce le risorse disponibili dividendoli tra più applicazioni, ma solo uno scheduler che si limita ad eseguire una sequenza di task secondo una politica FIFO preemptive basata su eventi.
- L'esecuzione del task corrente può essere interrotta solo se si verifica un evento (preemption). Un task non può interrompere un altro task .
- Un task è un contesto di esecuzione che viene eseguito in background fino al completamento e senza interferire con gli eventi del sistema.

Sistemi Operativi per Sensori : TinyOS

- Quando non ci sono task da eseguire lo scheduler mantiene il processore a riposo e attende la segnalazione di un nuovo evento per riprendere l'esecuzione.
- Come si intuisce lo scheduling ha due livelli di priorità, quello normale per i task e quello più alto per gli eventi che possono interrompere i task.
- TinyOS è scritto in nesC, una variante del linguaggio C, concepito per sistemi embedded e ottimizzato per la programmazione dei nodi sensori.

Sistemi Operativi per Sensori : TinyOS

- In TinyOS, il modello di comunicazione rappresenta una delle parti fondamentali.
- Lo scambio di dati tra i nodi utilizza gli active messages (messaggi attivi).
- Questo modello permette di inviare messaggi, a tutti o a un singolo nodo, tra i nodi vicini.
- Questa tecnologia è molto leggera, infatti non specifica dei meccanismi orientati alla connessione, ma ogni pacchetto e un'entità indipendente.
- Ogni *active message* contiene l'identificatore dell' *handler* (generalmente un intero che rappresenta il numero della porta) dell'application level che deve essere invocato sul nodo destinatario ed un payload dati nel quale sono specificati gli argomenti.

Sistemi Operativi per Ubiquitous Computing

- Palm OS è semplice, compatto, ma non implementa meccanismi di security.
- Symbian OS è più complesso, ma più generale e gestisce un efficiente multitasking.
- Windows Mobile supporta configurazioni flessibili e usa crittografia per la security.
- Embedded Linux offre l'interfaccia di programmazione avanzata di Linux su sistemi ubiqui di grana differente.
- TinyOS è il sistema operativo multitasking per sensori più diffuso. Di recente esistono sensori con SO Linux.
- Android è un sistema operativo e un ambiente di sviluppo di applicazioni open source per piattaforme mobili e pervasive.