

POLITECNICO DI TORINO

III Facoltà di Ingegneria dell'Informazione  
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Specialistica

# Ottimizzazione di applicazioni mobili per risparmio energetico



**Relatori:**

Prof. Maurizio Morisio

Ing. Paolo Falcarin

**Candidato:**

Luca Ardito

Febbraio 2010



## INTRODUZIONE

Molte funzionalità dei terminali elettronici moderni come processori ad alta velocità, display sempre più efficienti, dispositivi di memorizzazione di dati ottici/magnetici e moduli WiFi/GPRS/UMTS/HSDPA influiscono sensibilmente sul costo energetico dell'intero dispositivo portatile. Gli sviluppi della tecnologia delle batterie non riescono a stare al passo con la rapida crescita della richiesta di energia.

La batteria emerge quindi come un parametro chiave da controllare nella gestione del budget di potenza che si ha a disposizione per questo tipo di terminali.

E' necessario quindi trovare un buon compromesso tra prestazioni del sistema e durata della batteria, sia per quanto riguarda la fase di progetto, sia per quanto riguarda l'esecuzione del *software*, possibilmente avendo un'attiva partecipazione da parte dell'utente. E' quindi fondamentale incorporare lo stato attuale della carica residua della batteria nella strategia di ottimizzazione del suo ciclo di vita.

Per limitare al minimo l'attività di un'applicazione in modo tale da consumare meno energia possibile, bisogna rispettare i seguenti principi fondamentali:

- meno lavoro richiede meno energia: è necessario quindi:
  - modificare l'algoritmo in modo da rendere minimo il lavoro da compiere;
  - migliorare il compilatore in modo da introdurre ottimizzazioni basate sul processore ed eseguire controlli statistici;
  - preferire codice compilato e ottimizzato a *runtime* rispetto a codice interpretato;
- programmare ad eventi ove possibile: il polling va evitato;
- favorire la programmazione in ambienti *multicore*;
- non utilizzare timer periodici ad alta frequenza;
- introdurre scalabilità: l'applicazione deve poter ridurre le sue operazioni in situazioni di ridotta energia residua.

Le applicazioni di queste ottimizzazioni saranno indirizzate, nel corso di questa attività di tesi, verso telefoni cellulari di nuova generazione dotati di Sistema Operativo Android 1.5.

Il tipo di applicazioni analizzate riguardano i servizi di *Context Awareness* in fase di studio da parte del laboratorio di ricerca di *Telecom Italia*.

Per contesto si intende qualsiasi informazione che può essere utilizzata per caratterizzare la situazione di un'entità.

L'applicazione deve essere quindi in grado di combinare tutte le informazioni di contesto dall'ambiente circostante per descrivere la situazione corrente, determinare variazioni automatiche del suo comportamento oppure notificare all'utente l'avvenimento di un determinato evento. Questo tipo di applicazione deve essere spesso in esecuzione al fine di collezionare molti dati grezzi e di conseguenza esegue molte operazioni. Un'ottimizzazione del consumo energetico è quindi fondamentale per permetterne la diffusione e l'utilizzo. Si eviterà quindi l'esecuzione di operazioni ridondanti introducendo una modalità di funzionamento ad eventi.

Il *Local Context Broker* sviluppato per il sistema operativo Android (in futuro chiamato per semplicità *gLCB*) è composto idealmente da due livelli: il primo contiene il *Local Context Broker*, mentre il secondo i sensori.

Ogni sensore è installato come servizio ed è eseguito in *background* sul terminale.

## REALIZZAZIONE

A seguito delle criticità rilevate nella precedente versione di *gLCB*, si è reso necessario creare un meccanismo di start dei sensori, all'avvio di *gLCB*, che non fosse subordinato all'appartenenza dello stesso file eseguibile. Questo perché il numero di sensori è in crescita e la fase di test degli stessi non deve essere subordinata alla distribuzione di una nuova versione di *gLCB*.

Dopo aver apportato questa modifica i nuovi sensori possono essere a tutti gli effetti dei servizi indipendenti associati a file eseguibili differenti.

L'altra criticità della versione precedente di *gLCB* era la scansione sequenziale di tutti i sensori e la conseguente pubblicazione dei dati sul server. Il compito del server è quello di memorizzare tutte le informazioni collezionate dal terminale e di renderle disponibili ad altre applicazioni che vengono chiamate *Context Consumer*. Questo comporta la pubblicazione di dati duplicati ed un elevato consumo della batteria; l'intervento consiste nel permettere di dare al sensore la possibilità di eseguire una ricerca di un dato allo scatenarsi di un evento che identifichi un cambiamento di contesto. Ciò evita il *polling* ed introduce un meccanismo ad eventi. La tabella 1 elenca tutti i sensori disponibili e gli eventi che scatenano un update sul server.

Sensore	Eventi che scatenano un update
WiFiSensor	Avvio/Scadenza dato/Nuove WiFi rilevate
DeviceActivitySensor	Avvio/Scadenza dato
LocationSensor	Avvio/Scadenza dato/Spostamento superiore a soglia
DeviceStatusSensor	Avvio/Scadenza dato/Cambio modo d'uso
PhoneSensor	Avvio/Scadenza dato/Cambiamento cella
BluetoothSensor	Avvio/Scadenza dato/Nuovi device Bluetooth rilevati
DataSensor	Avvio/Scadenza dato/Cambio connettività (mobile o WiFi)
CallSensor	Avvio/Scadenza dato/10 chiamate (effettuate o ricevute)

Tabella 1. Eventi che scatenano una ricerca del dato di contesto da parte di *gLCB*

L'utente deve inoltre essere messo in condizione di scegliere se fornire informazioni dettagliate, a scapito del consumo energetico del terminale, o meno granulari ed avere una maggiore autonomia in termini di carica residua della batteria. Per questo motivo sono stati creati sette profili:

- VERY LOW
- LOW
- NORMAL
- HIGH
- VERY HIGH
- AUTO
- CUSTOM

Ognuno di essi fa variare il comportamento di *gLCB* in termini di ricerca e pubblicazione dei dati di contesto. In questo modo l'utente può scegliere il profilo che preferisce e, per mezzo del profilo CUSTOM, può decidere per ogni sensore la politica di update che ritiene più opportuna. Il profilo AUTO invece, seleziona il profilo attuale tra i precedenti in base alla carica residua attuale come descritto dalla tabella 2.

Intervallo	Profilo
Carica batteria collegato	PROFILO VERY HIGH
100% - 86%	PROFILO VERY HIGH
85% - 61%	PROFILO HIGH
60% - 41%	PROFILO NORMAL
40% - 16%	PROFILO LOW
15% - 5%	PROFILO VERY LOW
<5%	SPEGNIMENTO gLCB

Tabella 2. Funzionamento del profilo *Auto* di *gLCB 2.0*

Non avendo a disposizione un laboratorio specializzato per eseguire un'analisi dettagliata del consumo della batteria, si è misurato il tempo impiegato per scaricare completamente la batteria nelle due versioni di *gLCB*. Per quanto riguarda la seconda versione sono state effettuate le misure per ogni profilo d'uso. Per ogni diversa configurazione sono state svolte cinque ripetizioni e i valori riportati nel paragrafo successivo rappresentano il loro valor medio. Per facilitare l'operazione di misura dei consumi è stato creato un meccanismo che permette l'esecuzione di più cicli di test senza l'intervento da parte dell'utente. Ciò è stato reso possibile creando un caricabatteria controllabile via *software* (chiamato per semplicità *BatterySwitch*) che permette di caricare od interrompere la carica della batteria, per mezzo di un comando inviato direttamente dal cellulare. Questo sistema consente di fissare un intervallo di carica residua della batteria e di misurare il tempo impiegato ad esaurirlo. Si può impostare una coda di configurazioni da misurare in modo tale che, alla fine di ogni misurazione, il *device* pubblichi i dati ottenuti sul server ed inizi con la misurazione successiva. La figura 1 mostra *BatterySwitch* in funzione.



Figura 1. *BatterySwitch* interposto tra PC e *Samsung Galaxy*

Il terminale di riferimento è un *Samsung Galaxy* collegato alla rete tramite *HSDPA*. I risultati ottenuti evidenziano alcune problematiche che affliggono principalmente la batteria del terminale in quanto il dato di durata della batteria non si è rivelato stabile ed indicarne le cause non è semplice. Sicuramente il ciclo di vita della batteria stessa è da inserire tra le principali motivazioni in quanto, quando si eseguono queste misurazioni, bisognerebbe utilizzare una batteria nuova che lavora a temperatura costante. Ciò non è stato possibile per cui queste misure sono da considerarsi puramente indicative.

Gli scenari verificati sono i seguenti:

- terminale in standby con WiFi, Bluetooth e GPS abilitati (i test successivi prevedono che queste risorse siano attive);
- terminale con *gLCB 1.0* in esecuzione con ricerca dei dati di contesto ogni 180 sec.;
- terminale con *gLCB 2.0* in esecuzione con profilo *Very Low*;
- terminale con *gLCB 2.0* in esecuzione con profilo *Low*;
- terminale con *gLCB 2.0* in esecuzione con profilo *Normal*;
- terminale con *gLCB 2.0* in esecuzione con profilo *High*;
- terminale con *gLCB 2.0* in esecuzione con profilo *Very High*;
- terminale con *gLCB 2.0* in esecuzione con profilo *Auto*.

## CONCLUSIONI

Nella tabella 3 sono messe in relazione le informazioni sulle durate medie della batteria ed il numero di pubblicazioni medio per ora abilitando i vari profili, con la durata della batteria del *device* in *standby*.

Programma	Profilo	# update p/h	Durata media batteria	Differenza %
Standby	-	-	15 ore 43 min 16 sec	-
gLCB1	-	1,27	8 ore 23 min 32 sec	46,62
gLCB2	Auto	7,54	11 ore 41 min 33 sec	25,62
gLCB2	Very Low	1,25	13 ore 8 min 8 sec	16,45
gLCB2	Low	1,32	12 ore 7 min 35 sec	22,87
gLCB2	Normal	6,35	9 ore 13 min 6 sec	41,36
gLCB2	High	8,48	7 ore 55 min 55 sec	49,55
gLCB2	Very High	9,26	7 ore 46 min 30 sec	50,54

Tabella 3. Eventi che scatenano una ricerca del dato di contesto da parte di gLCB

La figura 2 mostra la sovrapposizione dei grafici che rappresentano il comportamento medio di scarica della batteria. I dati mostrano che il profilo *Auto* è di gran lunga il miglior compromesso tra pubblicazioni per ora medie e durata media della batteria. E' inoltre evidente che deve essere riconfigurato il profilo *Normal* in quanto sono presenti differenze troppo marcate in termini di durata media della batteria e di pubblicazioni medie per ora con il profilo *Low*.

Si possono formulare alcune ipotesi sul comportamento della scarica della batteria che verranno verificate con un nuovo task di misure. La durata media della batteria del Samsung Galaxy in *standby* è di 15 ore 43 minuti e 16 secondi. Questo risultato è evidentemente troppo basso in relazione ai dati dichiarati dalla casa. In compenso il dato di durata media della batteria utilizzando il profilo *Very Low* di *gLCB 2.0* è di 13 ore 8 minuti e 8 secondi: questo significa il 16,45% del tempo in meno. Si può mettere in relazione questo dato con la misura relativa alla vecchia versione di *gLCB* in quanto, analizzando il numero di pubblicazioni medio per ora (tabella 3) può essere considerato equivalente in termini di precisione dei dati di contesto pubblicati (per una misurazione eseguita con telefono cellulare fermo nello stesso punto). *gLCB 1.0* fa esaurire la carica residua della batteria con una media di 8 ore 23 minuti 32 secondi, che significa il 46,62% del tempo in meno. La riduzione del 30% è molto significativa.

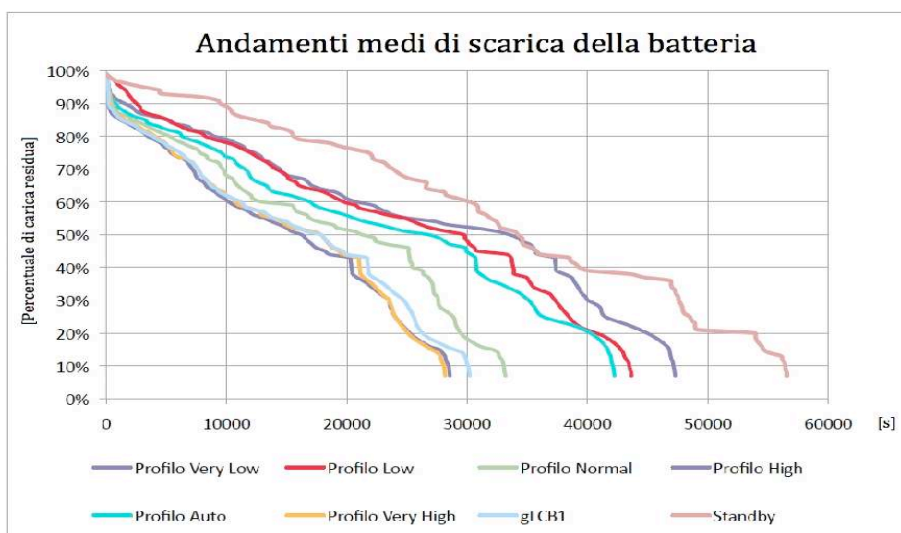


Figura 2. Sovrapposizione degli andamenti medi di scarica della batteria