Towards an Efficient Context-Aware System: Problems and Suggestions to Reduce Energy Consumption in Mobile Devices

Jose F. Mejia Bernal, Luca Ardito, Maurizio Morisio, Paolo Falcarin
Department of Automation and Information
Politecnico di Torino
Turin, Italy
{jose.mejiabernal,luca.ardito,maurizio.morisio,paolo.falcarin}@polito.it

Abstract—**Looking for optimizing the battery consumption is an open issue, and we think it is feasible if we analyze the battery consumption behavior of a typical context-aware application to reduce context-aware operations at runtime. This analysis is based on different context sensors configurations. Actually existing context-aware approaches are mainly based on collecting and sending context data to external components, without taking into account how expensive are these operations in terms of energy consumption. As a first result of our work in progress, we are proposing a way for reducing the context data publishing. We have designed a testing battery consumption architecture supported by Nokia Energy Profiler tool to verify consumption in different scenarios.**

Keywords-context-awareness; energy consumption; mobile device

I. INTRODUCTION

The increasing proliferation of mobile devices and the intention of defining universal standards related to mobile market, motivate many companies to implement context-aware standards, with the purpose of stimulating the fast and wide adoption of a variety of useful applications. The implementation of context-aware features results in benefits to the end-users, providing easy interoperability across operators and mobile terminals.

Many context-aware applications, based on different kind of technologies, also require new solutions for dynamically controlling device energy consumption. Such solutions can be visible by implementing a context-aware system, characterized by supporting context reasoning in mobile devices.

In order to extend context-aware functionalities for mobile devices, we have seen that the battery status is a relevant parameter involved in the device behavior. It is important finding an adequate synergy between the system performance and the battery life cycle, mainly related to the software execution and the active user participation.

A context-aware application has to be able to reduce its runtime operations in situations where the energy is considerably reduced. Reducing the device runtime operations means: taking at the minimum level the number of operations that have to be carried out by the system.

In order to limit at the lowest level the runtime operations to request the minimum energy possible, different kind of optimizations at hardware and software level, have been proposed to minimize the energy consumption for each useful computation.

Nowadays, operative systems for mobile devices provide different kind of APIs to check the battery state, and even business processes activities associated to battery state are updated according to the state changes.

Context data are any information that can be used to characterize a specific entity situation. A context-aware system has to be able to combine every context information related to the bounded environment in order to: (1) describe the actual situation and (2) determine automatic behavioural variations or notify the user about some specific event.

This system is constantly in execution in order to: (1) gather raw data and (2) execute different type of operations based on context reasoning.

Specifically, the proposed approach is based on a typical context-aware architecture. It implements features focused on reducing the energy consumption by avoiding execution of redundant operations.

On the other hand, different problems related to energy consumption in mobile devices are also analyzed, and a testing battery consumption mechanism is introduce to provide a reliable validation of a context-aware application, in terms of energetic efficiency.

The main purpose of our approach is to increase the adaptability of the system, by both identifying the main problems related to typical context-aware systems, and proposing feasible solutions according to the obtained battery consumption results.

The main features of our approach include updating context data procedures, according to eventual modifications derived from context information. All this with the scope of increasing the flexibility and improving the adaptability of the system.

The rest of this paper is organized as follows. Section II includes related works. Section III describes the main characteristics of Context-Awareness. Section IV describes the energy consumption system and its main features. Section V describes validation and results. Finally, Section VI includes conclusions and future work.

II. Related Works

In this section, we analyze and discuss some solutions that provide support for context-aware technologies. In [14] authors propose a context-aware battery management approach. The system proposed detects that the phone battery is low before the next charging opportunity is encountered.

Predictions algorithms are often used to establish when the next charging opportunity will be available, how much call-time will be required by the user, and how long the battery will last if the current set of applications continue to execute.

Some approaches are based on the next principles: (1) energy consumption should be predicted to allow devices to determine, how they should behave, according to scarce or plentiful energy, and (2) context information can be used to predict energy consumption policies.

Data from device's operating system (e.g. battery status, processes running on the device, calls made on the device) can provide important elements for reasoning and situation analysis. Additional context information (e.g. time of day, speed, presence of wireless devices) and prediction algorithms help to detect and infer specific situations.

An important metric that is not always considered in energy consumption is the battery age. It is important taking into account applications' battery usage as a way to provide adequate battery energy management.

There is not too much prior work on dynamic energy battery management. Some researches propose adaptation based on the logic and the content. Several researchers focus on the logic adaptation of services [12], the service is represented by components; and the adaptation is characterized by adding or replacing a component.

Other researchers [2] are focused on content adaptation; a typical example of content adaptation is changing the service presentation depending on the context data. The data properties can be modified in order to adapt the service according to terminal capabilities, network capabilities and/or even user preferences.

Prior research related to the limited battery lifetime problem is mainly focused on optimizing energy at different levels (e.g. hardware and application layer [1, 4]), including compiler-based energy optimizations [6].

On mobile devices, the interfaces that inform the user of the battery levels are not actually enough to provide a proper evolution of the capabilities of these devices for context reasoning.

There is some literature on predicting user location based on mobility traces [9, 8]. Battery lifetime research has also focused on analytical methods related to battery characteristics [13, 15].

There is also some work based on hardware power management for mobile computers [10]. Other works are related to a single component, such as network [7], disk [3], and CPU [11]. These results take mainly into account hardware components that are complementary to reducing energy usage through different kind of methodologies.

The main contributions of our approach provide both: (1) an analysis of the battery consumption behavior of a typical context-aware application, based on different sensor configurations and (2) a way to reduce the number context data publications. We have also designed a testing battery consumption architecture supported by Nokia Energy Profiler tool to verify energy consumption in different scenarios.

To sum things up, according to existing architectures, context-aware approaches are mainly based on collecting and sending useful context data to the server, without taking into account how expensive are these operations in terms of energy consumption.

The approaches mentioned before, differ from our approach mainly in aspects related to the consumption adaptation according to inferred situations [5]. Adaptation is only suggested when context information analysis reveals the need of applying changes.

III. Context-Awareness

Context information can be used to characterize a specific entity situation. It is divided into device context (e.g. net connectivity, communication cost and resources), user context (e.g. profile, geographic position, neighbors, social situation), physical context (e.g. temperature, noise level, light intensity, traffic conditions) and temporal context (e.g. day, week, month, season, year).

Combining every information obtained from the context retrieved, it is possible reconstructing the current situation. The characteristics of the surrounding environment determine the behaviour associated to mobile applications.

The set of states and parameters of the surrounding environment can determine a specific automatic variation related to the application behavior, or can allow the application informs the user about some specific event.

The Context-Aware concept is a paradigm that define how different kind of applications can discover the current situation, and consequently taking advantage of the context in which they are involved. Context-Aware applications use these features in order to perform reasoning operations and adaptation [16].

Context-aware computing relies on several independent enabling technologies, such as sensors for data input, hardware for data processing and artificial intelligence principles to extract rules by combining data and knowledge.

In order to develop a Context-Aware system, it is necessary considering the availability and the way in which the context information is used and processed.

A. *Context Data Level*

In this level, basic context data generated from several informative available sources, are retrieved and aggregated. This layer of abstraction collects context data in a fast and economic way, in order to integrate information sources and consequently supporting protocols and different kind of data formats.

B. *Context Analysis Level*

This layer identifies the user context and aggregates raw data with the aim of obtaining more relevant information. It also performs reasoning techniques on the information received to obtain new information with a higher abstraction level.

C. *Service Integration Layer*

Context-aware functionalities are exposed towards the service platforms. The actors involved in the Provider-Consumer model are: context provider, context consumer, context intermediary and context broker.

IV. ENERGY-AWARE CONSUMPTION SYSTEM

A typical context-aware client application is called Local Context Broker. Specifically, the local context broker is in charge of: (1) gathering context data retrieved from the device and (2) being the only contact point towards external/local components (e.g. data updating, data sending/receiving policies).

A. *System Architecture Analysis*

Typical Local Context Broker architectures are based on two levels: the first one is associated to the local broker, and the second one to the sensors layer. The local context broker provides an interface to allow other applications, installed in the device, request directly context data.

The Local Context Broker component is a background service that has as main function gathering, formatting and sending context information to external applications (e.g. context server).

Gathering context information is an operation performed by the sensors. Sensors are associated with different kind of information retrieved from the terminal (see Table I).

TABLE I. SENSORS DESCRIPTION

Sensor	***Description***
WiFi	WiFi nets
Cell	Which cell is connected to
Location	Geographical user position
DeviceInfo	Terminal information
DeviceSettings	Device configuration
Bluetooth	Bluetooth neighbors

B. *Suggested Features*

As a result of the critical problems found, the sequential scanning of every sensor and the consequent data publishing in the server, involves the publication of duplicated data and high energy consumption.

A typical Local Context Broker, often introduces some critical problems mainly related to: (1) high device battery consumption, (2) static data search and data publishing based on specific movement, (3) redundant context data transmission and (4) one single application controlling every sensor that collects specific context data.

In order to solve these critical problems, some modifications based on limited search and publishing are suggested. Context data search and publishing are expensive in terms of energy consumption.

The scanning of wireless nets, Bluetooth devices or geographic location, represents expensive operations. These operations have to be limited in cases where they are not needed.

It is necessary implementing a special control in the sensor, in order to both: (1) understand whether the data context retrieved is the same as in the previous scanning and (2) avoid the context data are published unnecessarily.

The sequential scanning of every sensor and the consequent data publishing mean duplicated data publishing and high battery consumption.

According to our analysis, the best solution is based on providing the possibility of executing a data search when it is perceived a specific event, that identifies a context variation. This feature is highly recommended because avoids the polling.

The Local Context Broker has to consider the context data expiration to handle this problem. It has to be implemented a timer mechanism included in every sensor that defines how long the data, in the server side, is valid.

C. *Update Policies*

The customer should be able to choose whether he wants to provide either detailed information, according to the device energy consumption, or less granular information in order to have a greater battery autonomy.

Some profiles could be defined in the server side to change the Local Context Broker behaviour in terms of context data searching and publishing.

V. VALIDATION AND RESULTS

Battery consumption analysis has involved the testing procedure related to how expensive are specific context scenarios. A local context broker, developed in Symbian (series 60 3rd Edition), has been tested.

Every configuration has been repeated twice. The reference device is a nokia n95 and the test configuration is based on “table II”.

TABLE II. TEST CONFIGURATION

Test Duration	10 minutes
Context Update	5 times
Clock	60 seconds
Test Cases	8
Use Cases	6

The battery life cycle has an important characteristic, when the mentioned tests are performed, it is relevant considering parameters related to how old is the tested battery and the temperature retrieved.

The battery consumption testing procedure is based on the next steps (see Fig. 1): (1) Energy profiler starts collecting data (e.g. voltage, current, CPU load), (2) TestGUI starts, (3) context sources are configured, (4) Local Context Broker starts, (5) Local Context Broker reads the configuration file and (6) TestGUI application collects data from Energy Profiler API.

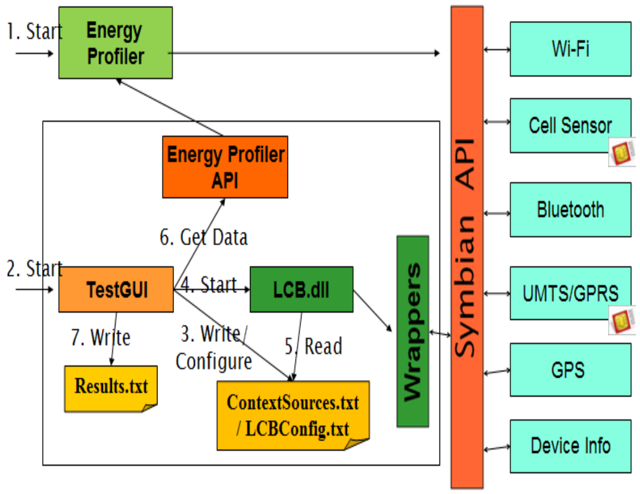

Figure 1. Testing procedure.

A. Use Cases

The use cases defined are the next:

- Indoor (Office): UMTS connection and not crowded place,
- Indoor (Office): Connectivity problems and not crowded place,
- Indoor (Classroom): UMTS connection and crowded place,
- Indoor (Classroom): Connectivity problems and crowded place,
- Outdoor (Courtyard): UMTS connection and crowded place,
- Outdoor (Courtyard): Connectivity problems and crowded place.

B. Findings

In order to establish the energy consumption behaviour, we have repeated the tests twice for both: (1) the terminal in standby and (2) the Local Context Broker running in background (see Table III).

Table III lists the battery consumption average in terms of how many times specific configurations are more expensive, in terms of battery consumption, than the device in standby conditions.

It is important considering that in the same column the consumption can increase more for some cells, it is because there are factors related to how much power the terminal used to connect to some specific UMTS cell.

The results show that the most expensive devices are Bluetooth, identified by Bt, and GPS. GPS is the most expensive sensor when the data transmission with the server is interrupted. When there are connectivity problems, the consumption is often higher than in normal connection cases.

TABLE III. TEST RESULTS

Use Case			Test Cases							
Indoor / Outdoor	UMTS Conn. / Conn. problems	Crowded / Not crowded	Standby	DevInfo	DevSet	Cell	Bt	GPS	WiFi	All
Indoor(Office)	UMTS Conn.	Not crowded	1,0	2,7	1,9	2,0	2,3	8,4	2,1	8,9
Indoor(Office)	Conn. problems	Not crowded	1,0	2,0	2,7	2,1	2,2	9,6	2,2	9,3
Indoor(Classroom)	UMTS Conn.	Crowded	1,0	2,4	2,5	2,4	3,0	7,7	2,6	10,0
Indoor(Classroom)	Conn. problems	Crowded	1,0	2,4	2,6	2,4	2,8	9,0	2,5	10,2
Outdoor(Courtyard)	UMTS Conn.	Crowded	1,0	2,4	2,5	2,5	2,9	8,8	2,8	9,6
Outdoor(Courtyard)	Conn. problems	Crowded	1,0	2,4	2,5	2,7	2,9	9,1	2,7	9,7

According to the test cases, the sensors ranking in terms of energy consumption, from the most expensive to the less expensive is: (1) GPS, (2) Bt, (3) WiFi, (4) DevSet, (5) Cell and (6) DevInfo.

According to the use cases results, in most of them, where there is not an adequate UMTS connection, the battery consumption is higher than the cases in which it is provided adequate UMTS connection. In crowded environments, the battery consumption of every sensor, except GPS, tends to increase compared to the not crowded case.

When all mentioned sensors are enabled at the same time, the current measure keeps symmetric for every test and is around 10 times greater than the case in which the Local Context Broker is disabled.

The consumption results do not consider factors related to: battery quality, temperature and which resources are used from every application. In order to reduce the energy consumption, it is be possible considering a different approach based on single resources management.

It is also important considering that in some cases the terminal could have transmitted more power to find an UMTS cell, in that case it is possible the final measure result may be greater than the logical expected value.

Since GPS and Bluetooth sensors are the most expensive, it is necessary reducing the number of context updates. GPS sensor is a component that has low connectivity for indoor situations, in this case reducing the number of context updates is a good mechanism to reduce the energy consumption.

DevInfo and Cell sensors revealed low consumption compared to the other sensors. The consumption associated to these sensors is almost the same for every situation. Since the DevInfo context data do not change often, the number of context updates have to be configured to the minimum permitted.

Not adequate UMTS connection cases revealed a higher battery consumption compared to the cases with optimal UMTS connection. A connectivity testing element is useful to avoid loss of data.

Server side policies to analyze context data and establish whether the context update can be reduced, according to data variations (e.g. cell changes, Bluetooth data, etc), are a good methodology for controlling consumption and applying solutions from the server side.

It is absolutely important and useful analyzing context data from the server side, in order to: (1) know whether the user change location and (2) avoid unnecessary updates.

Nowadays, many devices are equipped with accelerometer, this component is useful to estimate whether the terminal changed location, and implement policies to avoid the expensive usage of sensors associated to geographic coordinates, like GPS or UMTS cell connection.

VI. Conclusions And Future Work

The key goal of context-aware systems is providing relevant information and/or services based on current user context. In this paper, we analyze the battery consumption behavior of a typical context-aware application, running in background in a mobile terminal. This analysis is based on different sensor configurations.

We have designed a testing battery consumption architecture supported by Nokia Energy Profiler tool. We validated this tool by comparing the results retrieved from it, with the results retrieved from the oscilloscope. We have verified that the measures retrieved from the tool, match the measures verified in the oscilloscope.

It was necessary establishing the battery consumption in terms of how many times a specific configuration is more expensive than the terminal battery consumption in standby conditions.

We also discuss the main characteristics of context-aware approaches related to the battery lifetime problem, that are mainly focused on optimizing energy at different levels.

In particular, we analyzed some principles based on energy consumption to determine how mobile devices should behave according to scarce or plentiful energy, and how context information, can be used to infer energy consumption policies. These aspects are of importance for improving the efficiency of context-aware systems in terms of energy consumption.

Based on our findings, we propose a preliminary evaluation of the effectiveness of these context-aware architectures in terms of energy consumption for empirical studies.

There are still many relevant factors to improve the battery efficiency, mainly related to new available features (e.g. percentage of CPU usage, temperature and technology) in order to reduce the energy consumption.

As future work, we see many ways in which this work can be extended: (1) we would like to explore the robustness of our results across diverse platforms, in order to provide a bigger range of context-aware mobile applications studied. (2) we will implement and test a complete context-aware system, supported by the features suggested, in order to validate and confirm our findings.

Acknowledgment

This research has been partially funded by Telecom Italia S.p.A. / Research Projects within the collaboration with the Context Awareness group.

References

[1] 1 R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H. Yang, “The case for cyber foraging”, Proc. of the 10th workshop on ACM SIGOPS European workshop, 2002.

[2] 2 L. Boszormenyi, H. Hellwagner, H. Kosch, M. Libsie, S. Podlipnig, “Metadata driven adaptation in the ADMITS project”, in EURASIP Signal Processing: Image Communication Journal, Vol. 18, No. 8, Sept. 2003, pp. 749-766.

[3] 13 E. Douglis, P. Krishnan and B. Bershad, “Adaptive disk spin-down policies for mobile computers”, Proc. of the 2nd USENIX Symposium on Mobile and Location-Independent Computing, Ann Arbor, MI, April 1995, pp. 121-137.

[4] 3 J. Flinn and M. Satyanarayanan, “Energy-aware adaptation for mobile applications”, Proc. of the seventeenth ACM symposium on operating systems principles (SOSP 99), 1999.

[5] 15 L. Goix, M. Valla, L. Cerami and P. Falcarin, “Situation Inference for Mobile Users: a Rule Based Approach”, In workshop on Managing Context Information and Semantics in Mobile Environments (MCISME), May 2007.

[6] 4 T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, “Code transformations for energyefficient device management”, In IEEE Transactions on Computers, 2004.

[7] 11 R. Kravets and P. Krishnan, “Power management techniques for mobile communication”, Proc. of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 98), Dallas,TX, October 1998, pp. 157-168.

[8] 7 J. Krumm and E. Horvitz, “Predestination: Inferring destinations from partial trajectories”, Proc. of Ubicomp, 2006, pp. 243-260.

[9] 8 K. Laasonen, M. Raento, and H. Toivonen, “Adaptive on-device location recognition,” in Proceedings of the Second International Conference on Pervasive Computing, 2004.

[10] 12 J. Lorch and A. Smith, “Software strategies for portable computer energy management”. IEEE Personal Gommunications, June 1998, pp. 60-73.

[11] 14 J. Lorch and A. Smith, “Scheduling techniques for reducing processor energy use in MacOS”. Wireless Networks, October 1997, pp. 311-324.

[12] 5 B. Marquet, et. Al., “Secured services in a multi-tier architecture”, in World Telecommunications Congress (WTC 2002), Sept. 2002.

[13] 9 D. Panigrahi, S. Dey, R. Rao, K. Lahiri, C. Chiasserini, and A. Raghunathan, “Battery life estimation of mobile embedded systems”, Proc. of the The 14th International Conference on VLSI Design (VLSID ’01), 2001.

[14] 6 N. Ravi, J. Scott, L. Han, and L. Iftode, “Context-aware Battery Management for Mobile Phones”, Proc. IEEE PerCom '08, 2008, pp. 224-233.

[15] 10 P. Rong and M. Pedram, “Remaining battery capacity prediction for lithium-ion batteries”, In Conference of Design Automation and Test, 2003.

[16] 16 S. Schou, “Context-based Service Adaptation Platform: Improving the User Experience towards Mobile Location Services”, In ICOIN 2008: International Conference on Information Networking, January 2008, pp. 1-5.