Energy-aware Analysis of Synchronizing a Groupware Calendar

Fatos Xhafa, Daniel Palou
Universitat Politècnica de Catalunya
Barcelona, Spain
Email: fatos@cs.upc.edu

Leonard Barolli
Fukuoka Institute of Technology
Fukuoka, Japan
EMail: barolli@fit.ac.jp

Makoto Takizawa
Hosei University
Tokyo, Japan
EMail: makoto.takizawa@computer.org

Abstract—**With the fast development in Internet and Mobile technologies, we are witnessing a proliferation of applications that support online collaborative group work. Such applications offer many advantages as compared to traditional web-based online collaborative learning, the most salient feature being the ability to work not only geographically distributed but also on the move, anytime and anywhere. Such applications in the very recent past used to offer lightweight client applications, leaving the burden of the application logics to the server side applications. However, as mobile devices are each time more powerful in terms of computing and storage capacity, client applications at mobile device are each time more complex and heavy so that members of the group could be able to work both offline and online, which arises when members of the group use smartphones and can eventually run out of Internet connection from time to time, or simply want to develop some activities locally. This comes to the price of wasting the battery in shorter time and thus impeding the collaborative work for longer time intervals. Due to this reason, researchers and developers are interested to develop sophisticated mobile applications under energy-aware requirements. In this work we analyse the energy efficiency of a group calendar that offers 1) information sharing among all of members of the group; 2) synchronisation among local calendars of members and global group calendar; 3) conflict resolution through a voting system; 4) awareness of changes in the entries (tasks, members, events, etc.) of the calendar and 5) sharing of calendars among different groups. In particular, we analyse the energy efficiency due to the synchronisations mechanisms employed by a group calendar as synchronisation is among most compute intensive and time consuming tasks. Our study sheds light on the need of implementing various synchronisation mechanisms from lightweight synchronisation to full synchronisation and their applications according to energy profile of the mobile device.**

Keywords: Energy-ware computing, Battery life, Mobile devices, Calendar, Sharing, Synchronisation, Coordination, Groupware, Android, Mobile group work.

I. Introduction

With the fast development in Internet and mobile technologies, the collaborative group work is seeing a shift. Indeed, while in the recent past, collaborative group work was based on the web and peer-to-peer applications [16], [21], now most of the group work goes mobile. The use of mobile devices gives a new dimension to collaborative group work, which is not possible with Web-based group work, namely, the members of a group cannot only be geographically distributed, they can also do collaborative work *on the move*, anytime, anywhere [10] in both online and offline modes of work [22]. Therefore, collaborative group work using smartphones has become commonplace in most collaborative activities, including business collaboration in enterprises, online collaborative learning, collaborative group work at remote areas, disaster management scenarios, crowd-sensing collaboration, etc.

The design of a group calendar requires overcoming several challenges. Several research works have addressed the development of Android applications that support group work and can be integrated with enterprise information systems [6], [18]. For disaster management, Android applications can provide with timely support as shown in [7], [12], [15]. Several proposals for supporting collaborative work have been proposed for online collaborative learning to effectively scaffold learners, and particularly online groups of learners by smartphones and tablets applications [17], [19], [20], [23]. Data visualisation in a group calendar is also an important feature [5] for handling multiple schedules and highlighting common free times, especially relevant when the number of users is rather large and scalability is to be achieved.

Additionally, a collaborative group work calendar should meet several requirements such as:

1) sharing among all of members of the group;
2) synchronisation among local calendars of members and global group calendar;
3) conflict resolution through a voting system;
4) awareness of changes in the entries (tasks, members, events, etc.) of the calendar, and
5) sharing of different calendars among different groups

It should be noted however that as mobile devices are each time more powerful in terms of computing and storage capacity, client applications at mobile device are each time more sophisticated and can support complex tasks. This of course comes with the price of wasting the battery in shorter time and thus impeding the collaborative work for longer time intervals. Due to this reason, researchers and developers are interested to develop sophisticated mobile applications under energy-aware requirements.

In this paper, we analyse the energy efficiency of collaborative group work calendar that offers 1) information sharing among all of members of the group; 2) synchronisation among local calendars of members and global group calendar; 3) conflict resolution through a voting system; 4) awareness of changes in the entries (tasks, members, events, etc.) of the calendar and 5) sharing of calendars among different groups. In particular, we analyse the energy efficiency due to the synchronisations mechanisms employed by a group calendar as synchronisation is among most compute intensive and time consuming tasks. The objective of this study is to shed light on the efficiency of synchronisation mechanisms and to identify the range of energy waste corresponding to such synchronisation mechanisms from lightweight synchronisation to full synchronisation. We exemplify the approach for mobile devices under Android system and SugarCRM as a server.

The rest of the paper is organized as follows. In Section II we briefly describe the requirements for a group calendar supporting collaborative group work. The synchronisation mechanisms and its various types are presented in Section III. The design of the calendar, its implementation and deployment are presented in Section IV and the evaluation in Section V. We end the paper in Section VI with some conclusions and remarks for future work.

II. Requirements for energy aware mobile applications

Requirements for energy-aware applications are becoming commonplace in the design and implementations of mobile applications. As a matter of fact, for many authors, application-level energy efficiency requirements are considered as the next step in the evolution of power-efficient mobile systems as battery lifetime is a fundamental constraint. In Chen et al. [4], the authors argue that there is large space for energy saving on mobile applications based on application design, for which the authors propose a framework aiming to add an energy adaptation layer by providing a set of APIs and adaptation policies. Some authors [11], [14] exploit the offloading capabilities of executing certain tasks at server side. While there are several works on performance evaluation of Android systems (e.g. kernels [3]), particular applications need to be evaluated specifically in terms of the energy efficiency. In [13], the authors characterize the mobile applications according to their energy consumptions.

There are several requirements that a group calendar should satisfy to efficiently support collaborative group work. In terms of energy consumption, we could distinguish:

1) synchronisation among local calendars of members and the global group calendar;
2) conflict resolution mechanisms through a voting system;
3) awareness services to members about the changes in the calendar entries (tasks, members, events, etc.).
4) Calendar management (adding tasks, events, members, etc.)

The first of these requirements, namely the synchronisation, is among most time consuming, and hence, causing most energy consuming related to calendar application.

III. Synchronisation among local calendars of members and global group calendar

Synchronisation aims to keep consistent information at server side (the global group calendar) and at group members' sides (local calendars). When a member of the group changes some entry in the local calendar, this change should be reflected on the global calendar and all other local calendars of members affected by the change. The rest of members whose local calendar is not affected are notified about the changes. And, vice-versa, changes in the global calendar should be updated in local calendars of group members (see Fig. 1).

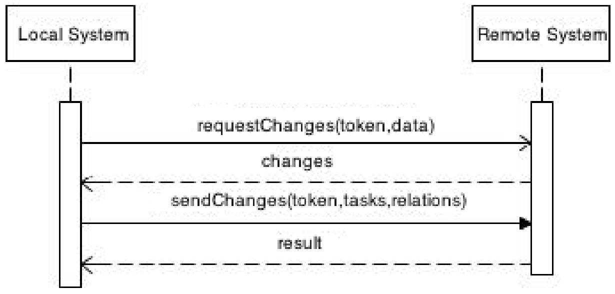
Local System
Remote System
requestChanges(token,data)
changes
sendChanges(token,tasks,relations)
result

Figure 1. Calendar synchronisation diagram.

There are many aspects that influence the synchronisation, such as decentralisation among members of the group (this issue has been much investigated for other models such as P2P Networks–see for instance [1]). By choosing to have the data spread across different devices, when a user makes a change to the data in its device, the change can be communicated directly to the other affected devices; or, the data can be stored centrally on a server and all the synchronisation is made through it. In the former case, synchronisation is more challenging due to the fact that the mobile devices can go on and off and data consistency is a real issue as it takes more time for the system to reach a consistent state when changes occur (usually *eventual consistency* is sought in this case). In the later case, using a server, it is easier and more reliable to manage the synchronisation and achieve desired degree of consistency but to the price of keeping the system centralised.

Among the various strategies for synchronizing data, the most used ones are full and incremental synchronisation.

A. Full synchronisation

The full synchronisation requires always all data to be synchronised. That is, if a device is to be synchronised with current server information, the server will always send all the data to the device (either in pull or push mode, according to parameter settings), and it is the device that should contrast that data received from the server against its data and update it accordingly, for example replace all old data by the newly received ones. This synchronisation can be seen as *safe mode synchronisation* in terms of synchronisation errors, because by receiving all the information of the server side we can be sure that the device will contain exactly the information of the server. Obviously, this synchronisation method implies a high cost of temporary data traffic, and it should be noted that part of this data traffic is redundant as the device has already part of it (usually small changes are done on the data locally). However, this kind of synchronisation is necessary when a user connects for the very first time to the system or when it connects after a long period of time. In other cases, an incremental synchronisation can be used, which is more efficient than full synchronisation. A diagrammatic view of the full synchronisation is shown in Fig. 2.

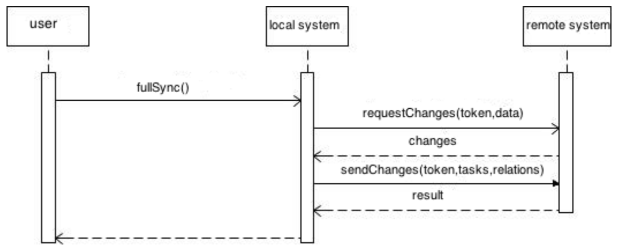
user
local system
remote system
fullSync()
requestChanges(token,data)
changes
sendChanges(token,tasks,relations)
result

Figure 2. Full synchronisation diagram.

B. Incremental synchronisation

The incremental synchronisation requires knowing only the data that has changed since the last synchronisation and the corresponding changes. Thus, a device receives only the changes occurred, not all information of the calendar. In this case, it is more likely to have a synchronisation error or a higher degree of inconsistency. For example, if for some reason the synchronisation fails, but through the process there was updated just the time/date of the last synchronisation, the synchronisation would be considered completed correctly. Clearly, this synchronisation is more error-prone in case of mobile devices but on the other hand it is much more efficient both in space and in data traffic since it only sends the necessary changes of the information, not redundant data. A diagrammatic view of the incremental synchronisation is shown in Fig. 3.

There is also possible to use both synchronisation modes, that is, the member of the group uses incremental synchronisation and from time to time, upon request, runs a full synchronisation with the server.

C. Conflict resolution

In every system that uses data synchronisation, conflicts may occur. In the case of the calendar, the most important

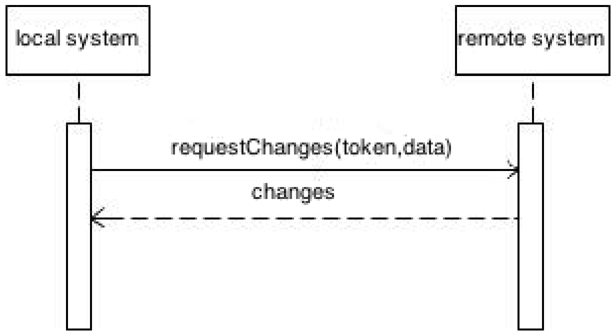

Figure 3. Incremental synchronisation diagram.

type of the conflict arises when 1) two users may be modifying the same data and 2) a change done by a member cannot be done persistent in the group calendar until the change is approved by the members of the group (in this later case, this requirement is important for some critical data such as deadline delivery, adding a new member to the calendar who is not member of the group, etc.)

There are many strategies to resolve conflicts [2], [8], one simple strategy is voting. In our case, and due to the importance of correct conflict resolution to the collaborative group work, the conflict resolution is left to the members choice, namely, the affected users take a decision by a simple majority vote. In case of a tie, the voting will be repeated.

D. Notification awareness services

This requirement is necessary to keep the members of a group informed on the group activity and to timely collaborate with other members. Awareness can take various forms such as availability awareness (availability of resources, availability of members, ...), context awareness (where the actions/changes took place), group process awareness (changes with regard to project workflow), etc. [16], [21]. Implementation of these awareness forms enables a timely and effective collaboration and coordination of online groups.

Awareness is implemented via event notification services. Events are defined and linked to Calendar, tasks, groups and users. Then, based on events, two types of notification services are implementing: a) event notification that requires user action and b) warnings (which are simply informative). For example, a user is notified that there is a pending task approval and is informed (*warning*) when a task is approved, completed and archived. A state diagram is defined for entities and events are linked to different states. It should be noted that awareness services take place whenever changes have occurred. The notification can be done in pull mode (configured by the user according to his preferences, for instance receiving notifications as soon as a change has occurred or receiving notifications in batches per time/day interval) or push mode (notifications are pushed by the server automatically to users). Obviously, the energy consumption depends on the notification strategy as it is more costly to receive notifications one by one than in batches or summaries.

E. Calendar management

The calendar management is done by the members of the group, besides the management of their own local calendar, who have access to the global group calendar. The calendar management includes creating a new calendar, adding members to the calendar, adding tasks and events to the calendar, making changes to existing entries in the calendar. etc. It should be noted that calendar management is not much a consuming task as compared to synchronisation because the management tasks are done sporadically and are bounded to a workflow of the group project. Some examples of the calendar managements (in diagrammatic language using the SugarCRM API) are *newCalendar* (see Fig. 4), *addMemberToCalendar* (Fig. 5), *addMemberToTask* (Fig. 6) and *confirmPendingApproval* (Fig. 7).

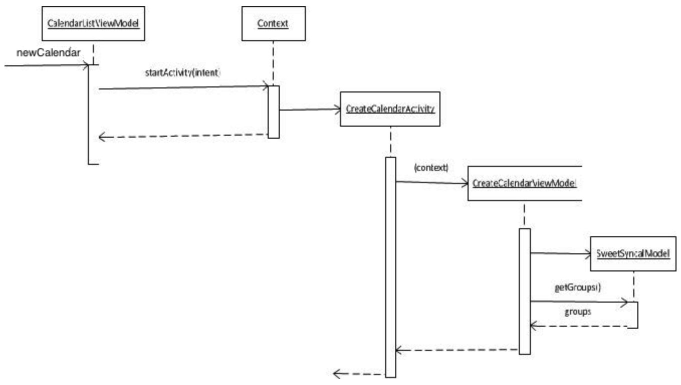

Figure 4. The sequence diagram of *newCalendar*.

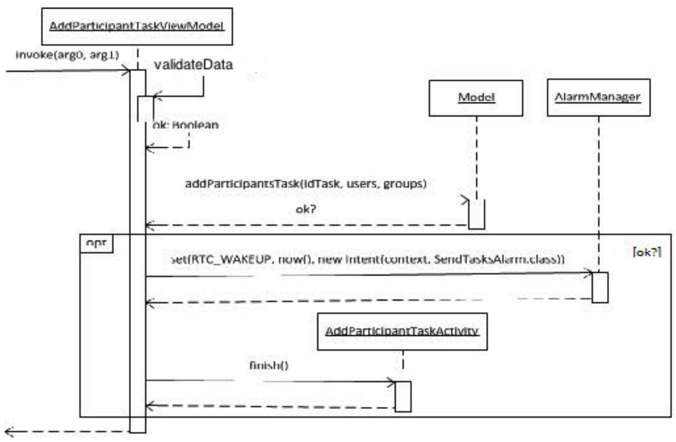
AddParticipantTaskViewModel
invoke(arg0, arg1)
validateData
ok: Boolean
Model
AlarmManager
addParticipantsTask(idTask, users, groups)
ok?
opt
[ok?]
set(RTC_WAKEUP, now(), new Intent(context, SendTasksAlarm.class))
AddParticipantTaskActivity
finish()

Figure 6. The sequence diagram of *addMemberToTask*.

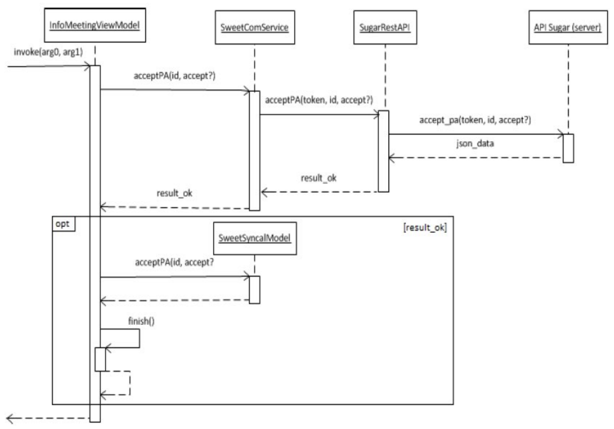

Figure 7. The sequence diagram of *confirmPendingApproval*.

IV. Implementation and technologies

A. Technologies

The following technologies have been used to implement the system.

Java: The system is designed to run on Android, whose programming language is Java.

SQLite: We use SQLite as a local database because the resources are quite limited. Storing data in files can also be considered, but it was decided to use the database as a more effective solution.

Android: Because we want our application to use the maximum number of available devices, it was decided that the application will be implemented under Android, nevertheless, the system can be fully implemented using other operating systems.

Android-Binding: We use this external open source library to achieve a greater independence between the view and our system using the MVVM pattern.

SugarCRM: SugarCRM is commonplace in applications for SMEs businesses. By choosing this application server, we simulate the case of a real project development in enterprises and make our application potentially useful for real setting.

PHP: The server functionality must be extended in order to adapt it to our system. Because SugarCRM is implemented in PHP, it is then the programming language used to expand it.

Memcached: Memcached is a distributed system that is used for storing cached data or objects, thus reducing the need to access an external data source.

JXTA: JXTA is a P2P open source platform by Sun Microsystems, defined as a set of XML-based protocols. These protocols allow devices connected to a network to exchange messages with each other regardless of the network topology. The definition of JXTA is abstract and based on a set of open protocols so that it can be brought to any modern programming languages. There are several implementations,

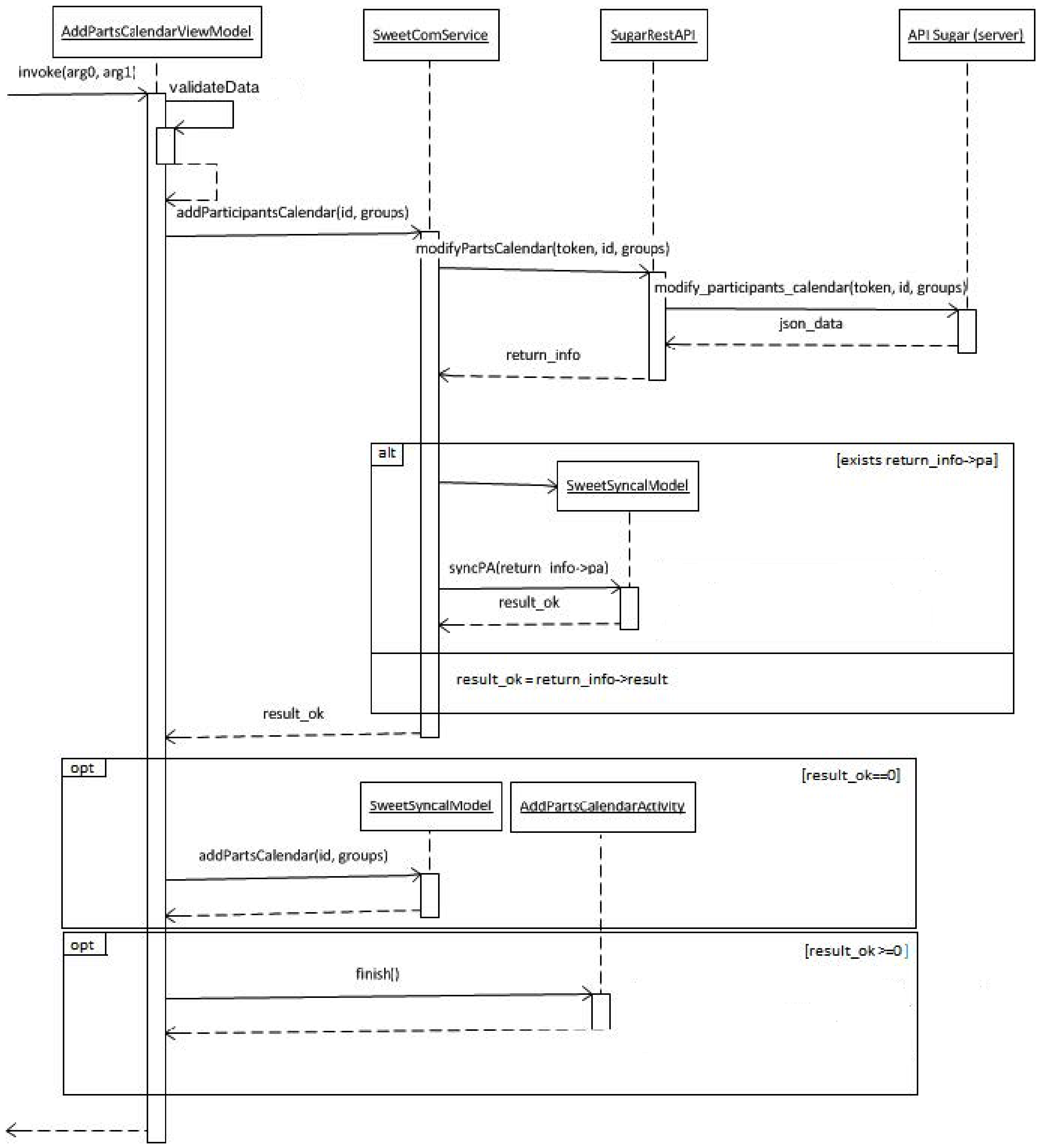

Figure 5. The sequence diagram of *addMemberToCalendar*.

the most advanced of which is JXSE the Java version. The version is called Java Micro Edition JXME. One important feature of JXTA is that it allows P2P communication even when peers are behind NATs or firewalls.

PeerDroid: PeerDroid is an open source library that implements protocols JXME for the Android platform. It allows to create applications for the Android platform using JXTA properties, creating a network of mobile and other traditional peers (computers, for example).

B. SugarCRM: modules vs. relationships

Besides analysing the algorithms, such as synchronisation ones, that impact on the calendar efficiency and thereof on the battery lifetime, in some cases the technologies used should be analysed as well. This is the case of using SugarCRM, a server application used in real life server applications.

SugarCRM uses modules and relationships to store information. *Modules* are similar to classes in a UML model, while *relationships* are the equivalent to associations. It should be noted that the relationships may store extra information, besides identifiers of the module instances they are linked to, but this extra information is rather difficult to be retrieved. The SugarCRM API offers a set of functions to retrieve and modify the module instances and relations but none of them is helpful to retrieve the extra information, and thus, the only way to get this extra information is directly from the database, which is not convenient.

The main extra information that we would be interested to retrieve from relationships is the modification timestamp in order to retrieve the changes of information from a certain timestamp for the needs of the synchronisation algorithms. The SugarCRM API offers the functions *get_modified_relationships* but it is useful only for the relationships User-Meeting, User-Call and User-Contact, where the timestamp corresponds to the second module Meeting, Call or Contact. It was observed therefore that SugarCRM API does not offer any functions to read the timestamp of modifications or filter information according to the timestamp, making thus more difficult the implementation of the synchronisation algorithms.

Summarizing, the modules and relationships have both advantages and drawbacks as follows.

Advantages: The relationships are more coherent with the model, avoid automatically repetition of associations and occupy less space in the DB. On the other hand, regarding modules, it is very easy to retrieve extra information, which is pretty much the same for all types of relations $1-1$, $1-^*$ or $^*-^*$.

Drawbacks: Regarding the relationships, recovering extra information is complex and its functions are more suitable for $1-^*$ relations, while for $^*-^*$ one needs to run $1-^*$ several times. Additionally, in recursive relations it is not possible to distinguish the extremes of the relation. On the other hand, as regards the modules, they are less coherent with the model, one has to manually check the existence of a module instance representing a relation, usually occupy more space in the DB, there cannot be defined foreign keys and require more reads for certain operations.

Given the advantages and limitations of modules and relationships, some entities such as *Voting* and *Reply* were to be implemented by modules. In fact in some cases, such as in the case of the entity *Precede*, the use of modules was mandatory because the relationship cannot distinguish the extremes of the relation for implementing the precedence constraints.

V. Deployment and evaluation

The system has been deployed at the RDLab infrastructure (http://rdlab.cs.upc.edu/, using SugarCRM server and has been evaluated according to different usability and performance criteria.

Several evaluation scenarios were designed and carried out using real mobile devices. The aim was to evaluate the efficiency of the proposed group calendar in terms of time, space and battery lifetime.

The efficiency of modules vs. relationships: With regard to the use of modules *vs.* relationships, the study showed that considering various scenarios of working with the calendar, using modules was more efficient (see Fig. 8).

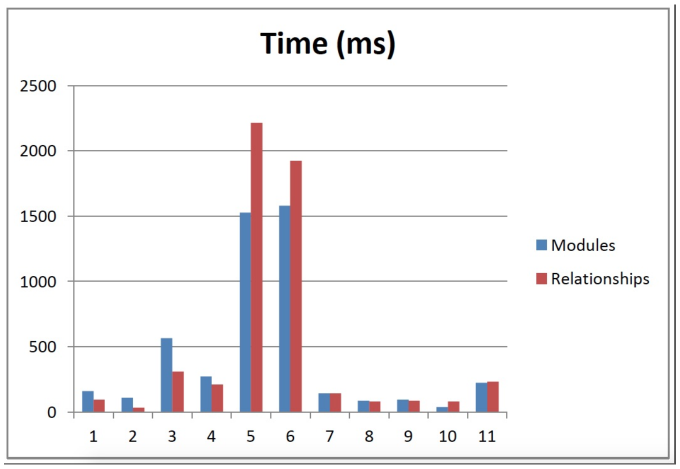

Figure 8. The time efficiency of implementation with modules *vs.* relationships.

The battery lifetime: SyncAdapter vs. Google Cloud Messaging: As discussed in previous sections, synchronisation can be done in various forms. For instance, it can be done by a pre-configured manner (through a *SyncAdapter*) every certain time interval, say every N minutes. This synchronisation mode, however, has the drawback that during that time interval the information in the calendar becomes inconsistent, especially for N large, should changes occur during that time. For shorter intervals of time, it might be useless to make synchronisation as in most cases there would be no changes in the calendars (locals or global calendar), which would imply a waste in the battery of the mobile device. It would be then desirable to have a more tailorable synchronisation.

In such situation, using an approach similar to Google Cloud Messaging (GCM), which allows to send messages to mobile devices from the server, seems interesting. Indeed, using this service, the mobile devices can send messages to the server to request synchronisation or vice-versa, from the server to the mobile devices to notify there are changes and thus synchronisation can take place.

We designed twelve use cases and measured the save in energy when doing the synchronisation via SyncAdapter *vs.* Google Cloud Messaging using the *GSam Battery Monitor* to measure the energy consumption. As can be seen from Fig. 9 for all the twelve tests, the GCM based synchronisation was much more efficient than SyncAdapter, namely it wasted *significantly less* battery during synchronisation process.

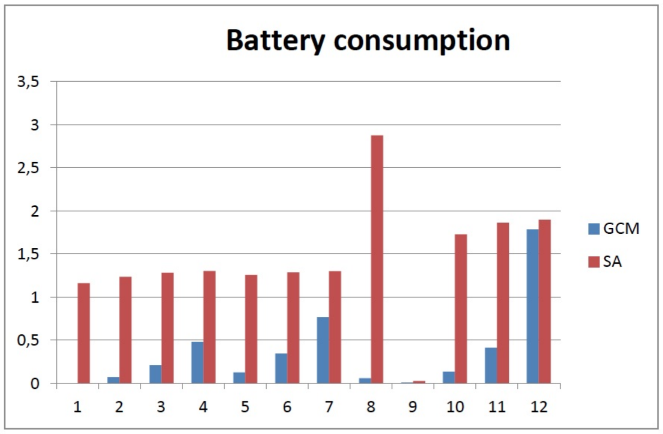

Figure 9. Battery consumption of synchronizing via SyncAdapter (SA) *vs.* GCM.

VI. Conclusions and Future Work

In this paper we have studied the impact on battery lifetime of mobile applications, specifically for a group calendar, which requires frequent synchronisations to keep the collaborative group work uptodate any time, anywhere. Indeed, many intelligent mobile applications currently support online collaborative group work but to the price of wasting the battery in shorter time and thus impeding the collaborative work for longer time intervals. We analysed the energy efficiency of collaborative group work calendar that offers 1) information sharing among all of members of the group; 2) synchronisation among local calendars of members and global group calendar; 3) conflict resolution through a voting system; 4) awareness of changes in the entries (tasks, members, events, etc.) of the calendar and 5) sharing of calendars among different groups. In particular, we analysed the energy efficiency due to the synchronisations mechanisms employed by a group calendar as synchronisation is among most compute intensive and time consuming tasks. We considered synchronisation via a SyncAdapter module, which can be preconfigured for synchronisation to take place at certain time intervals and via Message Services (similar to Google Cloud Messaging). Our study showed that synchronisation via Message Services consumes significantly less energy than synchronisation via SyncAdapter. In view of the results, this study stresses the need of implementing various synchronisation mechanisms from lightweight synchronisation to full synchronisation and their applications according to energy profile of the mobile device.

In our future work we would like to better use the battery life profile for synchronisation needs in order to find a suitable balance among the need for being synchronised according to remaining battery of the mobile devices.

References

[1] Achara, J.P., Imine, A. Rusinowitch, M. DeSCal - Decentralised Shared Calendar for P2P and Ad-Hoc Networks. *10th International Symposium on Parallel and Distributed Computing*. 223 - 231, DOI: 10.1109/ISPDC.2011.40, 2011

[2] Samuel A. Ajila, S.A. and Al-Asaad, A. Mobile databases – Synchronisation & conflict resolution strategies using SQL server. *IEEE International Conference on Information Reuse and Integration (IRI)*, 487 - 489, DOI: 10.1109/IRI.2011.6009598, 2011

[3] Luis Corral, L., Georgiev, A.B., Janes, A. and Kofler, S. Energy-Aware Performance Evaluation of Android Custom Kernels. *IEEE/ACM 4th International Workshop on Green and Sustainable Software (GREENS)*, 1 - 7, DOI: 10.1109/GREENS.2015.8, 2015

[4] Chen, H., Luo, B. and Shi, W. Anole: A Case for Energy-Aware Mobile Application Design *41st International Conference on Parallel Processing Workshops*, 232 - 238, DOI: 10.1109/ICPPW.2012.34, 2012.

[5] Engin, B., Cetinkaya, M., Ayiter, E., Germen, M. and Balcisoy, S. Maestro: Design Challenges for a Group Calendar. *12th International Conference Information Visualisation*, 491 - 496, DOI: 10.1109/IV.2008.91, 2008

[6] Gryaznov, G. and Kovin, R. Development of the cross-platform mobile framework for integration with enterprise information systems. *7th International Forum on Strategic Technology (IFOST)*, 1 - 4, DOI: 10.1109/IFOST.2012.6357607, 2012.

[7] Juhana, T., Widyani, R. N, and Mulyana, E. Mobile application for rapid disaster victim assessment. *7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 324 - 329, DOI: 10.1109/TSSA.2012.6366076, 2012.

[8] Kedia, A. and Prakash, A. Data synchronisation on Android clients. *IEEE International Conference on Communication Software and Networks (ICCSN)*, 212 - 216, DOI: 10.1109/ICCSN.2015.7296156, 2015.

[9] Kirda, E., Fenkam, P., Reif G., and Gall, H. A service architecture for mobile teamwork. In Proceedings of the *14th international conference on Software engineering and knowledge engineering (SEKE '02)*. ACM, New York, NY, USA, 513-518. DOI=10.1145/568760.568850 http://doi.acm.org/10.1145/568760.568850, 2002.

[10] Leonardi, L., Mamei, M. and Zambonelli, F. A Physically Grounded Approach to Coordinate Movements in a Team. In *Proceedings of the 1st International Workshop on Mobile Teamwork*, 2002.

[11] Marcu, M. and Tudor, D. Execution framework model for power-aware applications. *17th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, 1 - 6, 2011.

[12] Nakamoto, Y. and Akiyama, Sh. A Proposal for Mobile Collaborative Work Support Platform Using an Embedded Data Stream Management System. *35th IEEE International Conference on Distributed Computing Systems Workshops*, 23 - 28, DOI: 10.1109/ICDCSW.2015.16, 2015.

[13] Pinarer, O. and Ozgovde, A. Characterisation of mobile applications according to their energy consumptions. *22nd Signal Processing and Communications Applications Conference (SIU)*, 1995 - 1998, DOI: 10.1109/SIU.2014.6830649, 2014.

[14] Qian, H. and Andresen, D. An energy-saving task scheduler for mobile devices. *IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, 423 - 430, DOI: 10.1109/ICIS.2015.7166631, 2015.

[15] Rahman, K.M., Alam, T. and Chowdhury, M. Location based early disaster warning and evacuation system on mobile phones using OpenStreetMap. *IEEE Conference on Open Systems (ICOS)*, 1 - 6, DOI: 10.1109/ICOS.2012.6417627, 2012.

[16] Sapateiro, C., N. Baloian, P. Antunes and G. Zurita (2009) Developing Collaborative Peer-to-Peer Applications on Mobile Devices. Proceedings of the *13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009)*, Santiago, Chile, 2009.

[17] Djoni Haryadi Setiabudi, D.H., and Winsen, L.J.T. Mobile learning application based on hybrid mobile application technology running on Android smartphone and Blackberry. *International Conference on ICT for Smart Society (ICISS)*, 1 - 5, DOI: 10.1109/ICTSS.2013.6588081, 2013.

[18] Vojvodić, S., Zović, M., Režić, V., Maračić, H. and Kusek, M. Competence transfer through enterprise mobile application development. *37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 448 - 452, DOI: 10.1109/MIPRO.2014.6859609, 2014.

[19] Wei, S. The Design and Implementation of a Mobile Learning Platform Based on Android. *International Conference on Information Science and Cloud Computing Companion (ISCC-C)*, 345 - 350, DOI: 10.1109/ISCC-C.2013.136, 2013.

[20] Xhafa, F., Barolli, L., Caballé, S., Fernandez, R. Supporting Scenario-Based Online Learning with P2P Group-Based Systems. *NBiS 2010*:173-180, 2010.

[21] Xhafa, F. and Poulovassilis, A. Requirements for Distributed Event-Based Awareness in P2P Groupware Systems. *AINA Workshops 2010*: 220-225, 2010.

[22] Yang, Y., Supporting online Web-based teamwork in offline mobile mode too. *Proceedings of the First International Conference on Web Information Systems Engineering*, Vol.1, 486-490, 2000.

[23] Yang, H. Ch. and Wang, W.Y. Facilitating Academic Service-Learning with Android-Based Applications and Ubiquitous Computing Environment. *4th International Conference on Ubi-Media Computing (U-Media)*,191 - 196, DOI: 10.1109/U-MEDIA.2011.29. 2011