An Energy Efficient Implementation of Differential Synchronization on Mobile Devices

Jorg Simon
Know-Center
Graz, Austria
jsimon@know-center.at

Peter Schmidt
Mendeley
London, UK
peter.we.schmidt@gmail.com

Viktoria Pammer
Knowledge Technologies Inst.,
TU Graz, Austria
viktoria.pammer@tugraz.at

ABSTRACT

Synchronisation algorithms are central components of collaborative editing software. The energy efficiency for such algorithms becomes of interest to a wide community of mobile application developers. In this paper we explore the differential synchronisation (diffsync) algorithm with respect to energy consumption on mobile devices.
We identify three areas for optimisation: a.) Empty cycles where diffsync is executed although no changes need to be processed b.) tail energy by adapting cycle intervals and c.) computational complexity. We propose a push-based diffsync strategy in which synchronisation cycles are triggered when a device connects to the network or when a device is notified of changes. Discussions within this paper are based on real usage data of PDF annotations via the Mendeley iOS app.

Author Keywords

synchronization, collaboration, differential synchronization, energy efficiency, mobile phones

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: (A)Synchronous Interaction, Collaborative Computing

General Terms

Algorithms; Performance

INTRODUCTION

Differential Synchronisation (diffsync) [3] is a realtime synchronisation algorithm that compares two states of an item (e.g., a document), computes the differences and necessary changes in fixed intervals of time, and then applies changes. Diffsync is easy to implement compared to the quasi-standard operational transform (OT), mainly because client and server code are nearly identical, and synchronisation code is independent of user interface code. It is therefore "suitable for any content for which semantic diff and patch algorithms exist" [3]. As synchronous collaboration is increasingly mediated by mobile devices, energy efficient implementations of diffsync will gain in relevance.
Discussions in this paper are based on real usage data of editing PDF annotations within the Mendeley iOS app. Mendeley is a social document and content reference management tool. References, and their related PDFs can be shared amongst a group, and PDF annotations can therefore be edited by multiple users. The Mendeley iOS app uses the diffsync algorithm to synchronise PDF annotations.

MOBIQUITOUS 2014, December 02-05, London, Great Britain

DOI 10.4108/icst.mobiquitous.2014.257978

POTENTIAL FOR ENERGY OPTIMISATION

There are three areas in which diffsync can be optimised w.r.t. energy consumption for mobile devices, based on the underlying property of the unmodified differential synchronisation that the complete diffsync cycle is executed in regular, fixed intervals. Firstly, *empty cycles* are synchronisation cycles that are carried out even though no changes have been made to the document. Secondly, in 3G and GSM, *tail energy* is a medium power state after a network connection has been set-up. Depending on connection type and implemented protocol, the worst interval cycle for diffsync would be around $12s$ for 3G or $6s$ for GSM [1]. Thirdly, computational complexity directly correlates with CPU energy consumption [2]. Small changes have a high likelihood of being computationally less complex than large changes [4]. This argues for taking care to process small changes and small-sized items (the latter can be ensured by an intelligent choice of data structure).

PUSH-BASED ENERGY OPTIMISATION OF DIFFERENTIAL SYNCHRONISATION

The original diffsync paper [3] suggests adapting cycle intervals to current editing activities in a client between $1s$ and $10s$. However, this has been proposed in view of improving performance (e.g. processing speed, avoidance of merge conflicts), rather than energy consumption.

In order to strike a balance between computational performance and power consumption we propose to execute a diffsync only when changes occur, except for an initialisation cycle. Concretely:

1. In the initial state, the client connects to the network. In order to capture any changes that may have occurred in the meantime an initial diffsync cycle will be required.
2. When the client item is edited, the client initiates a diffsync cycle.
3. If a change arrives at the server, a push notification is sent to all clients. On receiving the notification clients execute a complete diffsync cycle.

Except at initialisation, no empty cycles are being carried out. This is the central property of the push-based optimisation of energy efficiency of diffsync. Note that reducing empty cycles only reduces energy consumption significantly for 3G and GSM connections, as empty cycles in WLAN drain the battery only minimally. At the same time, all changes are communicated to the server as fast as possible.
There are three possibilities how to identify that an edit has occurred on the client: Firstly, the "diff" part of the diffsync algorithm could be executed in regular intervals until a difference (an edit) is detected. Secondly, the data structure (e.g., file, database, in-memory) in which the application stores its content can be monitored for changes. Thirdly, it can be decided which user interaction means that an edit has occurred. Depending on the application and which granularity of edits needs to be identified, this may be as simple as noticing a user pushing a "save" button, or as complex as tracking every possible way of editing within a given UI.

Energy Consumption of Push Notifications on Mobile Devices
A network communication overhead is generated via push notifications. In both Android and iOS, push notifications are facilitated by a local service (on the mobile device) which regularly polls for push notifications (Google cloud messaging[1], Apple push notifications[2]).
We quantified this overhead in an experiment based on the Mendeley app on an iPhone 5s with a fresh install of iOS 7 and measuring battery drainage in $\%$ per minute: The local service that polls for push notifications was run for $2h$ without incoming push notifications (idle polling). Idle polling drains $\approx 0.02\%$ of battery power per minute in addition to the idle operating system. This is negligible. In a $2h$ run with a push notificaton every $6s$ (active polling), added energy consumption was 0.13% of battery power per minute. This is already a more sever battery drainage. Consequently, if changes occur regularly and frequently, it is more energy efficient to adapt cycle time to editing activity.

Minimum Time Between Push Notifications
When the interval between push notifications becomes smaller than $2s$, the mechanism became highly unreliable in our experiments, in the sense that the sequence of notifications was changed or notifications became lost. Server code should therefore take care to send notifications about changes only in intervals larger than $2s$. In addition, clients need to ensure that only one diffcycle is running at a time.

PUSH-BASED DIFFSYNC WITHIN MENDELEY

In the Mendeley app, synchronisation is on PDF annotations such as textual highlights and notes. The data content exchanged in each cycle is based on differences in position, and colour for highlights. The format of the exchanged data set is based on standard JSON. The Mendeley app's internal data structure for PDF annotations is a dictionary where every PDF annotation (highlight or note) is a dictionary entry identified by a unique ID. The "diff"-part of the diffsync algorithm therefore works on single dictionary entries neglecting the impact of computational complexity. Additionally, we could identify the occurrence of edits on the client by listening to modifications of the data structure. In Mendeley app's original diffsync implementation, a fixed $2s$ cycle interval was used.

Reduction of Energy Consumption in Mendeley
We analysed Mendeley usage data from a period of 4 weeks in May 2014. On average, users spent $431,4s$ ($\approx 7min$) within a PDF, reading and editing. The interaction time varies widely, with a minimum of $3,3s$ and a maximum duration of $1977,8s$ ($\approx 33min$). An overwhelming majority of diffsync cycles (96.8%) are empty and waste energy.
Assuming, a user stays $7min$ in a document on a 3G connection the original diffsync with a $2s$ cycle interval drains the battery 1.729% in these $7min$. The push-based diffsync however drains the battery only 0.084% in these $7min$ (again, measurements were on an iPhone 4s with a fresh install of iOS 7), thus using only $\approx 5\%$ of the original energy.

CONCLUSION

We have proposed and discussed a push-based optimisation of differential synchronisation. For the Mendeley application, given typical app usage, this energy-efficient diffsync implementation leads to significant improvements. On a more general note, we emphasize that energy optimisations of differential synchronisation should be done based on knowledge of a collaborative system's usage data.

ACKNOWLEDGMENTS

The Know-Center is funded within the Austrian COMET Program under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

REFERENCES

1. Balasubramanian, N., Balasubramanian, A., and Venkataramani, A. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09*, ACM Press (2009), 280–293.
2. Damasevicius, R., Ziberkas, G., Stuikys, V., and Toldinas, J. Energy Consumption of Hash Functions. *Electronics and Electrical Engineering 18*, 10 (Dec. 2012), 81–84.
3. Fraser, N. Differential synchronization. In *Proceedings of the 9th ACM symposium on Document engineering - DocEng '09*, ACM Press (2009), 13.
4. Myers, E. W. AnO(ND) difference algorithm and its variations. *Algorithmica 1*, 1-4 (Nov. 1986), 251–266.

[1] `http://developer.android.com/google/gcm/index.html`
[2] `https://developer.apple.com/notifications/`