Towards Mobile Energy-Adaptive Rich Internet Applications

Johannes Waltsgott
Faculty of Computer Science
Technische Universität Dresden, Germany
johannes.waltsgott@tu-dresden.de

Klaus Meißner
Faculty of Computer Science
Technische Universität Dresden, Germany
klaus.meissner@tu-dresden.de

***Abstract*—Composite web applications built from reusable components are replacing traditional, monolithic Rich Internet Applications (RIAs). Based on the rising number of smartphones and the increasing usage of mobile applications, composite web applications arrive on mobile devices. While the computing power of the latter rapidly grows, an unsolved problem persists: the limited energy resources of mobile devices. We propose an architecture for mobile energy-adaptive RIAs, which allows for energy optimization by adapting the distribution of components between the server and the device and minimizing the data communication.**

***Keywords*-composite applications, mobile applications, energy efficiency, component migration, mashups**

I. Introduction

Service-oriented architectures are current practice to build reusable and agile composite applications from loosely coupled services and resources. Lately, this composition paradigm has been deployed to the presentation layer as well. This paved the way for mashups, or *composite web applications*, as an alternative to former, monolithic RIAs. Mashups have gained acceptance for consumers as well as enterprises [1].

At the same time, the number of smartphones sold has risen tremendously. A Gartner survey shows that, compared to the same period of 2010, smartphone sales increased by 42 % in the third quarter of 2011, with an even higher estimation for Q4 and early 2012 [2]. Not only has the number of devices been growing, but also the usage of mobile applications. Today, emailing, web browsing, personal navigation and social media applications are used by many smartphone users. In the future, new usage scenarios will arise, making excessive use of the device's sensors [3]. In parallel, mobile applications rely heavily on remote data and cloud storage to overcome limitations regarding storage on the device and to support collaborative scenarios. However, there exists one thing, which does not even rudimentarily keep pace with the increasing distribution, performance and usage of smartphones: the device's battery capacity (cf. [4]). Since mobile devices are generally required to last as long as possible, the limited energy budget and severe energy consumers are ongoing issues for smartphone users.

In this paper, we introduce our approach towards mobile energy-adaptive RIAs, in which we capitalize on the mobile communication management at application layer. We focus on composite applications based on CRUISe [5], a universal composition platform for mashups.

The paper is structured as follows. In Section II, we give a brief overview of the CRUISe composition platform. Section III summarizes the challenges and related work regarding energy-aware mobile applications. In Section IV, we introduce our proposal and describe its respective parts. Finally, we discuss our findings and outline future work in Section V.

II. The CRUISe Composition Platform

Our approach towards energy-efficient RIAs is based on the CRUISe Platform for universal mashup composition, whose principles have been introduced in [6]. CRUISe extends the known service-oriented paradigm to include the presentation layer. Applications consist of universal parts, which provide data access, business logic and UI. These CRUISe *components* share a generic component model and a platform-independent description language, the *Semantic Mashup Component Description Language* (SMCDL, see [7] for more details). The inner workings of a component are encapsulated by an interface consisting of three abstract concepts, namely *property*, *operation* and *event* (cf. [8]). The public state of a component is represented by its properties, while changes of the inner state result in publishing events, which could be consumed by the runtime or other components. The functionality provided by a component is accessible by calling its operations. This allows for a loose coupling of components, where an event-based communication architecture routes event messages from publishing components to the respective subscribers via *event channels*.

A composite application is described by a generic *composition model* [8], referencing the involved components' IDs, the required event channels and layout information. A service-oriented infrastructure supports the dynamic execution of CRUISe applications at runtime, as depicted in Figure 1. The composition model (upper left of figure) is interpreted by a CRUISe *runtime environment*, shown in the middle, that brings the composition to life, using universal components provided by the service layer at the bottom.

The runtime environment receives the component code of every component from a *component repository*, shown

 ISBN: 978-1-61208-200-4

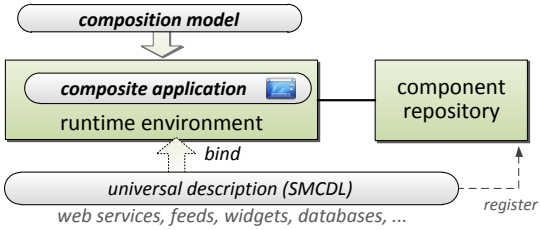

Figure 1. Architectural overview of the CRUISe composition platform [6]

on right, where all components are registered. Finally, the runtime integrates and instantiates the components and establishes the specified event channels.

Thus far, CRUISe provides a mature platform for universal mashup composition, which allows for dynamic component integration at runtime and is already in practical use in industry. However, the existing capabilities of the CRUISe runtime lack support for energy-aware execution of CRUISe applications. This is of high importance, especially when used on mobile devices, as motivated before.

III. Challenges and Related Work

First of all, it is crucial to determine the main energy consumers of current smartphones. Carrol and Heiser performed a detailed analysis of a mobile phone's power consumption [9], identifying the display and graphics as the main consumers, followed by the GSM radio while the CPU, generally, is of lower relevance. They profiled phone components isolated by several benchmarks and measured the energy consumption for various usage scenarios, e. g., audio and video playback, text messaging, phone calls, emailing and web browsing. The results show that the display (LCD panel, touchscreen, graphics and backlight) head the power consumption of all none-GSM-intensive scenarios. Otherwise, e. g., phone calls, emailing and web browsing, the GSM radio respectively the WiFi system consumed the most of power, while WiFi showed a noticeable higher energy efficiency for data transfer. Since the backlight of the display is either automatically dimmed by the operating system or explicitly set to a user-specified value, optimizing a device's communication behavior provides the most promising approach for energy optimization at application level.

A specialized analysis of energy consumption of mobile communication (GSM, 3G and WiFi) was performed by Balasubramanian et al. [10]. Their results show a fair energy efficiency for smaller data size (10 KB) using GSM and for bigger data size (>100 KB) using WiFi. 3G consumes significantly more energy for data transfer, according to the high *tail energy*, which covers the energy consumed while remaining in a high power transmitting state even after the actual data transfer is completed. Based on their measurements, they derived a power model for all three communication technologies covering ramp, transition and tail energy as well as the tail time. Besides, they proposed *TailEnder*, a network protocol to be integrated in mobile operating systems, which schedules data transfer for delay-tolerant applications or prefetches data (e. g., search results) for suitable usage scenarios. It has to be surveyed, whether TailEnder could be used in addition to our approach, since it is a very data-centric view. Besides, its suitability for highly interactive RIAs has to be proven yet.

Based on the understanding of mobile communication as relevant energy consumer, it is of high importance to influence the communication behavior and distribution of mobile applications. MAUI [4] is an approach to optimize the device runtime by energy-aware distribution of mobile code. It relies on special attributes in the code, marking methods to be (potentially) executed on remote hosts. A profiler collects context information on the device, the network and the program state at runtime. A solver processes the information and decides whether a method should be invoked locally or remote, taking the overhead (transfer time and cost, processing cost) into account. Since we focus on component-based RIAs, where the components act as black boxes and are handled by their interface description only, MAUI seems not suitable.

A more coarse-grained approach proposes a method for energy-efficient workflow distribution [11], in which a workflow model is enhanced by a network model and a data model. The network model describes valid environments (mobile or hosted) for a workflow's activity. The data model describes the transmission costs between two activities, which are derived from the data size to be transmitted and the power model presented in [10]. The most efficient distribution of the activities is calculated by a *minimal cut algorithm*, applied to a *cost graph*. Afterwards, the workflow is deployed accordingly. Their evaluation showed average energy savings up to 37 % for optimized distribution. However, since their approach focuses on workflows with determined size of data transmitted between activities, it does not suit highly interactive RIAs. Moreover, we strive for energy optimization at runtime rather than at application deployment.

Flinn proposed remote execution for mobile applications [12], focusing on energy-aware adaptation of application quality to optimize energy consumption by delivering application performance to meet user requirements. However, he did not account for communication costs while distributing application code but concentrated on network bandwidth and latency as performance parameters only.

In summary, it can be stated, that relevant energy consumers in current smartphones, which could be controlled by system or application level, have been identified clearly: the mobile communication devices. Thus, research focuses on optimizing mobile communication: from fine-grain code level to coarse-grain approaches. Nevertheless, there are shortcomings regarding energy optimization for highly interactive, component-based RIAs, whose communication behavior could not easily be predicted prior to runtime.

 ISBN: 978-1-61208-200-4

IV. Proposal for Solution

Facing the afore mentioned drawbacks, we introduce *energy-adaptive Rich Internet Applications* (eRIAs): an architecture and runtime environment for composite mobile web applications, which allows for energy-efficient reconfiguration of applications and migration of components between client and server. It consists of three main parts, as depicted in Figure 2: (1) a *context monitor*, collecting

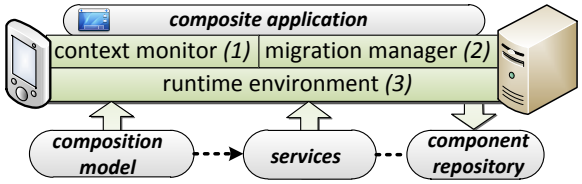

Figure 2. Architectural overview of the eRIA proposal

information on the application and device state and on user requirements at runtime; (2) a *migration manager*, deciding, which reconfiguration of a component distribution between server and mobile device is the most energy-efficient one for a given context and migrating the affected components accordingly; (3) a *runtime environment*, supporting the dynamic execution of application components on the server as well as on the client. In the following, we discuss the respective parts of our proposal in detail.

A. Context Monitor

CRUISe-based composite web applications consists of UI and service components, loosely coupled by an event bus. Components are encapsulated by an interface, describing *properties*, *operations* and *events* by the SMCDL. Thus, their inner state is unknown to the runtime environment, which requires monitoring their behavior externally. The Context Monitor shall collect information on the component's communication traffic to allow for a calculation of the energy cost. Communication between components can be measured at the Event Manager (EM), which is part of the CRUISe runtime environment (cf. [13]), since the EM is responsible for wiring components by delivering event messages to operation calls according to the definition of the composition model. Communication with external services can be measured at the Service Access (SA, also part of the CRUISe runtime environment), which acts as a global proxy for external requests due to client-side security restrictions (cf. [13]). The SA delivers the received data via a given callback method to the component.

Besides, the Context Monitor shall gather information on the energy context of the mobile device. These information involve the current battery state, whether the device is just charging, what kind of mobile communication technology is used and basic parameters for signal strength and bandwidth. Finally, the Context Monitor shall be able to collect user requirements, e. g., a user forces a high-performance mode of an application approving a higher energy consumption knowing he will soon be able to recharge the device.

B. Migration Manager

Based on the information collected by the Context Monitor, the Migration Manager derives energy costs for data communication. Given the data size and the specific communication technology used, the required energy for a transmission can be calculated with the energy model presented in [10]. Based on the composition model, which describes all integral components and their type (UI / none-UI), the Migration Manager identifies which components could be migrated in general. At this time, we assume, that UI components remain unchanged on the mobile device and will not be replaced adaptively.

Analyzing the calculated energy costs for communication and the given device's context information as well as the user requirements, the Migration Manager determines which components have to be migrated to or from the server. The component's state has to be serialized, transferred to the server, de-serialized and the component has to be instantiated with the former state on the server. To lower the transfer overhead between client and server, the component's code itself is not moved to the server. Instead, the server fetches the component code from the repository via the component's ID, known from the composition model. If a component should be migrated from the server to the mobile device, the savings of the communication costs must exceed the transfer costs of the component and its state, if the component has not been instantiated on the client earlier. Thus, the Migration Manager holds information on the components' migration history. Main parts of the Migration Manager shall run on the client to avoid transmitting great quantities of context data to the server for processing.

C. Runtime Environment

The runtime environment is responsible for interpreting the composition model, contacting the component repository, receiving the component code, integrating the components and establishing the needed event channels between the components according to the composition model. Implementations of a CRUISe runtime environment have been developed for server or client side only, recently as a Thin-Server-Runtime [13], running completely within the web browser. However, our approach requires a runtime environment on both the server and the client side, which allows for the dynamic execution of none-UI components on both sides. Thus, a dynamic *Client-Server-Runtime* is currently under development, which provides component integration and instantiation on both the server and the client as well as an event bus (embedding the needed event channels) between server and client to allow for communication among migrated components. Initially, we will utilize a server-side JavaScript executor to run CRUISe components (which

 ISBN: 978-1-61208-200-4

typically are JavaScript-based) on the server, supporting further platforms in the future. Access to external services (cf. SA) will be provided in the same manner on server and client, to make the actual location of execution transparent to the components.

V. Conclusion and Future Work

In this paper, we have introduced our proposal for mobile energy-adaptive RIAs briefly. Based on collected context information on the application and the device at runtime, the Migration Manager decides whether to migrate components between server and client to minimize data transfer and, thus, to save energy and lengthen the device's uptime.

We neglect energy optimization on server side within our approach, as we focus on mobile devices. Optimizing energy consumption of component-based applications on servers has been studied within the CoolSoftware project [14], introducing *Energy Auto Tuning* [15].

Our approach has also some limitations. Due to the focus on communication behavior, CPU-intense applications with minor external communication will not benefit much from this proposal. Moreover, it could be difficult to derive a migration strategy for components that consume and publish data of equal quantity, as they will cause high communication costs regardless of where they are executed. Frequent migration of components could result in high overhead costs. This can be addressed by migration policies and initial distribution suggestions for components, derived from experiments run prior to application deployment.

To achieve our aim, we will face the following challenges next: We will survey, whether the Context Monitor could also use prediction technologies (besides runtime monitoring) to determine data traffic or if methods and work from machine learning could be useful as well. Further research is required with regards to migration strategies, clarifying where components should be integrated initially: on the server or on the client, as this impacts the initial data traffic. Finally, we have to complete the implementation of the dynamic Client-Server-Runtime.

To evaluate our approach, we plan a representative user study, which allows us to measure energy consumption for several usage scenarios and communication technologies on current smartphones and to compare our solution with classical mobile RIAs.

Acknowledgment

The work of Johannes Waltsgott is founded by the German Federal Ministry of Education and Research under promotional reference number 13N10782.

References

[1] V. Tietz, S. Pietschmann, G. Blichmann, K. Meißner, A. Casall, and B. Grams, "Towards Task-Based Development of Enterprise Mashups," in *Proc. of the 13th Intl. Conf. on Information Integration and Web-based Applications & Services*. ACM, 2011, pp. 325–328.

[2] Gartner Inc., "Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011," 2012, Mar 22th. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1848514

[3] M. Satyanarayanan, "Mobile Computing: the Next Decade," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, no. 2, pp. 2–10, 2011.

[4] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *Proc. of the 8th Intl. Conf. on Mobile Systems, Applications, and Services*. ACM, 2010, pp. 49–62.

[5] CRUISe Consortium, "CRUISe project," 2012, Mar 22th. [Online]. Available: http://www.mmt.inf.tu-dresden.de/cruise/

[6] S. Pietschmann, "A Model-Driven Development Process and Runtime Platform for Adaptive Composite Web Applications," *Intl. Journal On Advances in Internet Technology*, vol. 2, no. 4, pp. 277–288, 2009.

[7] CRUISe Consortium, "CRUISe Mashup Component Description Language MCDL," 2012, Mar 22th. [Online]. Available: http://www.mmt.inf.tu-dresden.de/cruise/mcdl/

[8] S. Pietschmann, V. Tietz, J. Reimann, C. Liebing, M. Pohle, and K. Meißner, "A Metamodel for Context-Aware Component-Based Mashup Applications," in *Proc. of the 12th Intl. Conf. on Information Integration and Web-based Applications & Services*. ACM, 2010, pp. 413–420.

[9] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *Proc. of the 2010 USENIX Annual Technical Conf.* USENIX, 2010, pp. 21–34.

[10] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *Proc. of the 9th ACM SIGCOMM Conf. on Internet Measurement Conference*. ACM, 2009, pp. 280–293.

[11] D. Fischer, S. Föll, K. Herrmann, and K. Rothermel, "Energy-efficient Workflow Distribution," in *Proc. of the 5th Intl. Conf. on Communication System Software and Middleware*. ACM, 2011, pp. 2:1–2:8.

[12] J. Flinn, "Extending Mobile Computer Battery Life through Energy-Aware Adaptation," Ph.D. dissertation, Carnegie Mellon University, December 2001.

[13] S. Pietschmann, J. Waltsgott, and K. Meißner, "A Thin-Server Runtime Platform for Composite Web Applications," in *Proc. of the 5th Intl. Conf. on Internet and Web Applications and Services*. IEEE, 2010, pp. 390–395.

[14] CoolSoftware Consortium, "CoolSoftware project," 2012, Mar 22th. [Online]. Available: http://www.cool-software.org/

[15] S. Götz, C. Wilke, M. Schmidt, S. Cech, and U. Aßmann, "Towards Energy Auto Tuning," in *Proc. of 1st Annual Intl. Conf. on Green Information Technology*, 2010, pp. 122–129.

 ISBN: 978-1-61208-200-4