Context-, Resource-, and User-Aware Provision of Services on Mobile Devices

Andre Ebert, Florian Dorfmeister, Marco Maier, and Claudia Linnhoff-Popien
Mobile and Distributed Systems Group
Ludwig-Maximilians-University
Munich, Germany
Email: {andre.ebert, florian.dorfmeister, marco.maier, linnhoff}@ifi.lmu.de

***Abstract*—Today's smartphones are equipped with numerous different sensors and are capable of providing a large number of diverse services. But due to the high energy consumption of built-in sensors, as well as because of the energy consumed while analyzing gathered raw data, a bottleneck in resource supply is likely to occur during a service provision. This bottleneck leads to one of the biggest challenges regarding the developments on mobile devices: the trade-off between a high short-term service performance and sustainable energy management. Furthermore, despite of numerous hardware improvements (e.g., energy saving displays, batteries with bigger capacities, etc.) this issue remains unsolved. Hence, software-based approaches can be used to optimize the resource and energy management on mobile devices according to the user's preferences, existing context information, and the current energetic state of a device. Moreover, a holistic energy management enables the system to provide context dependent services.**

In this article, we present a concept for custom tailored service provision on mobile devices in combination with EMMA (Energy Management Middleware Architecture), a modular architechture for managing a mobile device's service infrastructure in relation to its current resource state as well as to the user's individual preferences. Additionally, we give an insight into our prototypical application, which demonstrates EMMA's core concepts including its featured approach of individual service provision.

Keywords–Mobile resource management; User-centric service provision

I. INTRODUCTION

Depending on the usage intensity, the batteries of today's smartphones do not often last longer than for one or two days. Reasons for that are high performance hardware components leading to high energy consumption, extensive sensor usage for data acquisition, improperly developed applications, or others. With this being a known problem, several hardware- and software-based approaches for optimizing the energy consumption on mobile devices have been developed. However, typically, only single components of a mobile system are subject of optimization, while there are no known approaches covering all existing subdomains and the corresponding possibilites for saving energy. Furthermore, due to the fact that most services on a mobile device can be provided by more than one specific procedure, opportunities for using a granular performance and quality concept are facilitated. Disregarding known hardware-based improvements, this leads to the necessity of an integrated, software-based energy and service management approach in order to protect the limited resources on a mobile device.

But improving a device's energy consumption is not only about extending its batteries' life span. Additional goals are the increase of the user's usage experience and the usability of an energy management system by adjusting specific services to individual user needs and requirements. In order to achieve these objectives, a user's preferences, current context information and the device's energetic system state, as well as currently active services must be considered [1]. Furthermore, an energy management system must be able to identify and prioritize services and functionalities which are important or critical to the user. At its best, such a system is able to present these services to the user with a dynamically tailored Quality of Service (QoS) whenever they are requested. QoS is described by individual parameters for each service, depending on its specific output. E.g., the QoS of a data transfer service could be described by its transfer speed and the amount of data to be transferred, the one of a positioning service by its accuracy and the time span needed for acquiring a location.

In order to describe the user's context and to be able to react to a present situation, context modeling is required first. According to Dey, "Context is any information, that can be used to characterize the situation of an entity [2]. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Relying on this definition, it is possible to identify the kind of raw data which is needed for context modeling and to specify sources and procedures for its acquirement. In the following, we present four categories of sources for determining raw context data.

1) **Conventional Sensors:** Data about the users environment, specific activities or current whereabouts can be acquired with built-in smartphone sensors such as Global Positioning System (GPS), gyroscope or accelerometer.
2) **Communication Interfaces:** Network technologies like Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS) or Long Term Evolution (LTE) plus Bluetooth and Wireless Local Area Network (WLAN) can also be used for determining the user's location in different granularities [3][4].

3) **Media Data Analysis:** Data like taken pictures or surrounding noises can also be analyzed in order to extract information about the user's situation, e.g., concerning current moods or a visited location [5].
4) **User Data Mining:** Personal user data like emails, played media files or calendar entries could be valuable for the process of extracting context information. Its analysis can provide knowledge about whereabouts, important dates like birthdays, or upcoming appointments.

However, using some of the presented approaches for the acquisition of context information can be very costly. As shown in Section II, all of them consume energy during the process of raw data gathering, as well as for the procedure of extracting information from it. Consequently, the energy consumption caused by context acquisition always has to be compared to the savings enabled through context-aware energy allocation. Besides, there are also privacy issues to be considered. Especially in 3) and 4), sensitive user data, which may need additional protection by privacy policies, is used for analysis.

We are not aware of any concept for an integrated energy management system, which satisfies all of the aforementioned needs. However, there are a lot of ideas for partial improvements of single components. In the following, we specify the requirements for a holistic system and introduce EMMA, an energy management middleware architecture which considers the dynamic and modular integration of existing energy improvement concepts as well as controlling the device's service provision [1]. Furthermore, it monitors the system's energy status and adapts it according to the current context, active services and the user preferences.

In this section, we presented an introduction to energy management matters on mobile devices as well as pointing out the necessity of providing custom tailored and context-dependent services to the user. In Section II, we provide a brief analysis of a smartphone's energy consumption in order to understand which components are responsible for draining its battery and where improvements can be achieved. Based on a literature survey and existing approaches for energy management on mobile devices, as well as on context-aware mechanisms for optimizing energy consumption, we present the requirements for an extensive context-aware and user-centric mobile energy management architecture in Section III. Eventually, the concept and implementation of this approach, coined Energy Management Middleware Architecture (EMMA), will be described in Section IV. A particular emphasis is laid on the provision of requested services in dependence of the system's current resource state as well as on the user's personal preferences and performance claims. After offering some insights into a prototypical implementation in Section V, a conclusion and future work are addressed in Section VI.

II. Related Work

In order to optimize the energy consumption on a mobile device, as well as using information about its context and energetic state for custom-tailored service provision, one first needs to understand which of a device's components or applications is draining most of its resources. Later on, it is possible to contrive ideas and optimization concepts based on this knowledge, as well as on already existing optimization and provisioning approaches.

A. Individual measuring of energy consumption

There are two different techniques for measuring energy consumption on smartphones. One relies on software based functions provided by the operating system; the other one uses an external power meter. Manweiler et al. describe the installation and usage of a Monsoon Power Monitor for external power consumption measurement [6]. However, this approach does not pay attention to the individual consumption of the device's components since it only identifies the overall consumption. Examples for software based and combined measurement concepts are eProf [7], PowerTutor [8], and WattsOn [9]. They all make use of predefined energy profiles describing the energy consumption of specific device components and indicate, that network and sensor usage, CPUs, displays and media playback are the main consumers of energy. Additionally, Pathak et al. [7] examined the energy consumption of different popular applications like Facebook, Angry Birds and others and figured out that 65-75% of the consumed energy is used by third-party advertisement modules. Furthermore, they name typical bugs in operation systems, which are responsible for the dissipation of energy.

B. Optimization concepts

In the following, we review different optimization concepts concerning sensors and data transfer. Furthermore, the usage of prediction and data mining in order to prioritize, schedule and optimize tasks on mobile devices is examined.

1) Sensing optimization: There are several approaches for lowering the energy consumption of mobile devices based on optimizing sensor usage. Adaptation of QoS parameters is one attempt that can be used to achieve this goal. In particular, the trade-off between service quality and consumed energy is a relevant matter in this context. Besides the adaptation of a service's performance, there is also the possibility of substituting sensors. In this case, specific sensors are substituted by other technologies which are capable of providing a comparable service while lowering the energy consumption [10][11]. If a complete substitution is not possible or needed, there is also the way of combining sensors, e.g., by triggering approaches [12]. Here, the acquisition of data from one sensor is triggered only when a second sensor reaches some kind of previously defined threshold. An example for this is the combination of a smartphone's accelerometer and its GPS sensor in the SenseLess concept introduced by Abdesslem et al. [13]. Their approach is to activate the GPS-Sensor only if the accelerometer detects the user being in motion. For evaluation, a user is equipped with two smartphones, one running the standard iOS positioning methods in a 10 second interval, the other one working with the SenseLess algorithm. The results showed, that SenseLess needed 85.5% less energy than the standard approach. The determined locations differed from 0.4m to 41m with an average value of 8m. But because

the energy consumption was determined on basis of the battery level, there is cause for inaccuracies through wrong interpreted up-and-down-movement, as well as due to the systems general energy consumption. As shown by Priyanta et al. [14], the combination of different sensors is to be used with caution and after individual evaluation. In their case, the computing of determined accelerometer data, which was gathered in a 10-minutes time interval with a frequency of 4Hz to 6Hz, consumed more energy than the location determination via GPS for 5 minutes, determining one location fix each minute. Schirmer et al. compared conventional GPS positioning approaches to custom triggering and polling concepts on iOS 4.3.1 and furthermore rated them concerning their energy consumption as well as their positioning accuracy [15]. With the polling approach, the authors tried to lower the energy consumption on a mobile device by the usage of a fixed sensor data request frequency. Thus, unnecessary data acquisition due to oversampling is about to be obviated. The triggering approach facilitated the usage of a device's built in GPS sensor in combination with its accelerometer. Therewith, the GPS usage was paused as long as no user movement was detected in order to save the device's resources during that time. For evaluation, Schirmer et al. recorded positioning fixes for a predefined route, including some predetermined movement breaks, with all of the three positioning concepts. The analysis of the result showed, that the performance of the polling was comparable to the standard iOS procedure. With the help of the triggering approach, the positioning procedure's energy consumption could be reduced, even though the positioning fixes' quality was significantly worse. Chon et al. introduce SmartDC, which not only aims at lowering the energy consumption while determining the users location, but also tries to predict the user's future positions and important places [16]. In order to accomplish that, the authors use unsupervised learning techniques, mobility prediction, and prediction of system usage based on Markov models. Using this approach, energy savings up to 81% and a prediction accuracy up to 80% have been reached. The maximum delay time was 160 seconds.

A complete substitution of sensors is used for Ambience-Fingerprinting [17], a technique that is especially suitable for indoor location determination, where no GPS signal can be received. Therefore, raw data of different kind is checked for specific attributes, so called fingerprints. An early concept of Pirantha et al. enables the user to discover his position via ultrasonic and radio fingerprints within a cm-accuracy [17]. Azizyan et al. present SurroundSense, a system for location determination on basis of environmental fingerprints like the spectral composition of noises or visual signatures [18]. The authors also analyze the cumulation of existing fingerprinting techniques, such as motion, noise, acceleration, brightness, color, and spectral contents of WLAN signals. In an evaluation with four users in predefined positioning clusters, 93% of all positions could be determined correctly. However, the consumed energy was not compared to the consumption of standard Software Development Kit (SDK) methods.

2) Optimizing data transfer: Due to the extensive energy consumption of data transfer and communication technologies, this field provides a lot of possibilities for achieving energy savings. 2G, 3G, and 4G networks each use a different amount of energy for different tasks. Hence, already by choosing the best suited technology for each task, the overall energy consumption can be lowered. For example, 2G networks are suited better for calls than 3G networks, but 3G networks are much more efficient for data transfer [19]. Perucci et al. conclude that – even though additional power is consumed by the handover process needed for changing networks – significant energy savings can be achieved by consistently switching to a 2G connection for calls [20].

Furthermore, both Balasubramanian et al. [19] and Falaki et al. [21] reveal that not only the overall size of the transferred data is responsible for the consumed energy, but also the size of single transferred packages: Especially, small packages cause a transfer overhead up to 40%. Moreover, the Transmission Control Protocol (TCP) requires multiple transfers of the same packages due to package loss. To this end, Falaki et al. propose a bigger server side buffer and an optimization of the usage rules for the transfer networks in order to solve these problems, predicting energy savings up to 35%.

Another transfer-related optimization approach is TailEnder, provided by Balasubramanian et al. [19]. TailEnder divides applications into two groups, i.e., applications that tolerate delays in data transfer and those, which can profit from data prefetching. Based on this classification, data is loaded in bundles to minimize the device's staying in the high energy state of the IEEE 802.11 standard. Additionally, TailEnder makes usage of the advantages of the different transfer networks. In comparison to other approaches, 60% more newsfeeds and up to 50% more search results for web requests could be processed while consuming the same amount of energy.

3) Using History and data-based prediction: The analysis of user preferences and interaction patterns, as well as the usage of data mining techniques on historic or context data can reveal precious insights concerning the energetic regulation of a mobile system. For example, calendar or appointment specific data can be used to predict a user's current and future whereabouts. Predictions about the device's future energy level, combined with the information about upcoming tasks can be used to adapt the energy balance oriented to tasks rated critical by the user [22]. Oliver et al. succeeded in predicting the energy level of mobile devices correctly for a time slot of one hour with an error rate of only 7%. Within 24 hours the error rate was 28%. In order to achieve that, they classified the gathered data of more than 17,300 BlackBerry users and clustered it afterwards [23][24].

Trestian et al. conducted a network-based study which should reveal relations between the user, his movement patterns and used applications, in order to predict common interaction patterns. Their results indicate that the usage of specific applications is significantly related to the user's current movements or his location [25].

III. Requirements

As shown in Section II-B, there are numerous approaches, which address single optimizations for specific system components or services. None of them tries to wrap all existing

concepts in one architecture, which organizes them in a modular way and provides requested services with an adequate output quality. In order to create a holistic energy management system which is capable of providing custom tailored services to a user, the following requirements for such an architecture can be identified:

1) **Universal validity and responsibility:** The energy management system is solely responsible for managing all resources of the system and the access to them. It receives all service and resource requests and answers them in an adequate and energy efficient way.
2) **Context and user awareness:** Typical context data like the current time and location, upcoming tasks, individual user preferences or social relations, as well as the systems current energetic state are used to provide services in a customized manner respecting the user's situation and whereabouts.
3) **Modularity:** A holistic energy management system is comprised of a multitude of different components, each responsible for different subtasks. In order to benefit from existing and future ideas in each of these areas, functionality is encapsulated in only loosely coupled modules, which can be altered or replaced without affecting the rest of the system.
4) **Scalability and extensibility:** New energy optimization concepts or services can get installed in an easy way, the user is able to use improved modules without non-trivial update processes.
5) **Definition of generic interfaces:** To make the integration of new modules possible, a definition of generic interfaces oriented on the lowest common denominator of installable modules is necessary.
6) **Adaptive data collection:** If data collection is required (e.g., sensor data), the system selects a service that acquires data adaptively to given preconditions. This may be the system's current energy state, the service quality as demanded by the service requester or forecasts concerning needed energy for future tasks. Furthermore, collected data needs to be cached for later usage.
7) **Task prioritization:** In order to take the user's preferences concerning critical tasks into account, the system is able to adhere to task priorities and assign more privileges to higher prioritized tasks.
8) **Clarity and comprehensibility:** The architecture must be structured in a clear and comprehensible way. This is essential to enable developers to integrate new optimization concepts or module packages into the existing system.

IV. EMMA - AN INTEGRATED MANAGEMENT APPROACH

Based on the requirements identified in Section III, we now present a holistic approach for managing resources in a user-centric and context-aware way, coined *EMMA (Energy Management Middleware Architecture)* [1]. Furthermore, EMMA allows an easy integration of existing and future energy optimization concepts. It provides its services in an adaptive, context- and preferences-aware manner, controls service parameters and monitors the system's energy state. EMMA runs continuously in the background and is the exclusive interface to the system's resources (such as sensors) and services for other applications. Thus, EMMA is able to coordinate resource demands of all applications and to fulfill their requests in the most energy-efficient way.

A. Main components

EMMA consists of two main components, namely, the Control Unit and the Service Provider. These and their subcomponents are depicted in Figure 1. In the following, the components are described.

1) Control Unit: The Control Unit is the central component dedicated to respond to service requests, select appropriate services based on context, preferences and demands, and monitor services and the system's energy state. If required, it adapts service parameters to ensure execution of critical tasks. In order to make intelligent decisions, the Control Unit employs information obtained from sub-components such as the Energy Manager, the Schedule Manager and the Service Identificator. Service requests, responses, and communication with the Service Provider, as well as among the subcomponents themselves are handled by a dedicated Controller. The Energy Manager analyzes the energy status of the whole system, calculates energy consumption of individual tasks and tries to predict future energy levels. These results can be combined with information about charging opportunities in vicinity or in near future, to reserve energy for future tasks or to preserve execution of critical tasks by performance adaptation for the longest time possible. The Service Identificator selects the most appropriate service for any given service request from the whole set of available services – as provided by the Service Provider – based on service parameters and quality of service characteristics. In general, service selection algorithms are geared towards a specific subset of services, since service parameters vary for different kinds of services. In order to monitor active and upcoming services, the Schedule Manager keeps track of currently running and future tasks, as well as respectively associated information.

2) Service Provider: The Service Provider is a passive component which does not contain any logic or executive power. Its core task is the integration and administration of service modules, as well as the provision of all requested services, the registration of needed callbacks and the caching of sensor data. To achieve that, the Registry Manager scans the folders containing all installed modules and corresponding manifest files, in order to identify added or removed modules for installation or removal. Afterwards, the current system state is saved in the Registry Cache, which provides these information for module identification matters to the Control Unit. As soon as a module was identified and released for usage by the Control Unit, the Service Provider creates a new service object and integrates the identified module in order to use its unique functionalities. The service is accessible via specific interface methods (see Section IV-D).

In order to avoid redundant acquisition of raw data, the Service Provider caches sensor data or other requested information

and provides it, if needed. In case that granularity, accuracy, and recency of cached data comply with the demanded QoS-parameters contained by an incoming service request, it can be answered by returning this data. The advantage of this procedure is obvious: By responding to requests with cached data, the instantiation of a new service object can be avoided and existing resources can be preserved for upcoming tasks.

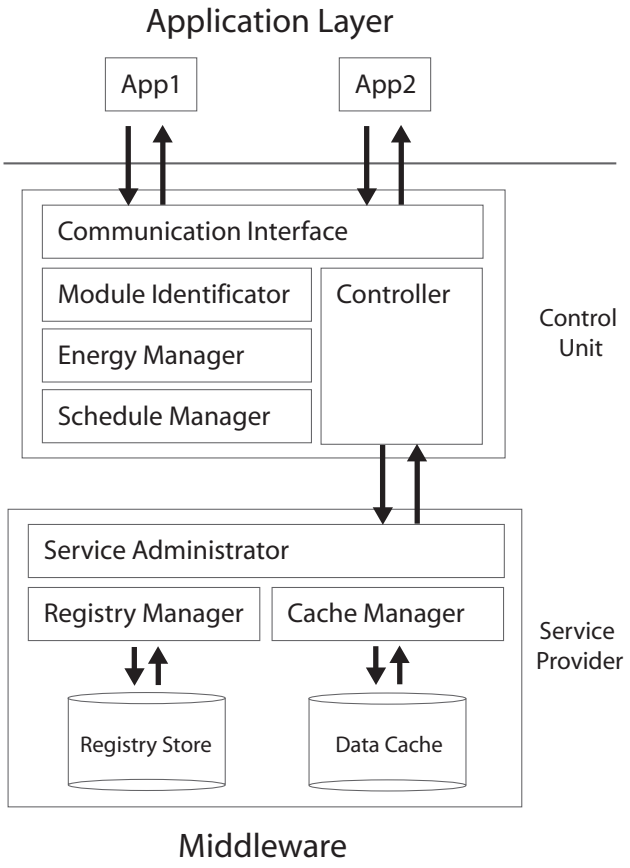

Figure 1. Overview of the EMMA concept and its components.

B. Modular service concept

A core feature of EMMA is its modular service concept. A service is defined as any functionality, that a user or an application might request, e.g., data transfer, acquisition of sensors data or media playback (see Section IV-C). In order to maximize the efficiency of providing these services, EMMA uses a highly modular concept. To this end, for each functionality a service class is defined in a formal way. Each service class might use different technologies to provide a requested service and the selection of a specific module might depend on diverse request properties, e.g., on the demanded service quality or the system's currently available energy. The internal logics of these different technologies are wrapped completely transparent in a service module and it is neither visible to requesting applications, nor the EMMA architecture itself. The number of possible service modules per service class is unlimited, and the logic that is wrapped inside the module package is completly up to the module developer. The only thing, which needs to comply with EMMA's development guidelines, is the proper implementation of the predefined interfaces of the parent service class, as well as the delivery of an associated manifest file. The manifest contains information about the modules QoS parameters, as well as its parent service class and must be transcripted in a standardized way.

Later on, after the best suited module for a specific use case was identified (see Section IV-E), it is integrated into a service object at runtime (see Section V-B). Apart from the growing diversity of available technologies, this simple plug-and-play integration is another advantage of EMMA's modular concept. It enables optimized logic and improved technologies being integrated without the necessity for a complete system update (see Section V).

C. Requesting services

In order to answer an incoming service request in an adequate manner, several processes need to be executed. This is due to the necessity of considering the user's context and preferences, as well as the most recent energetic system state. Figure 2 illustrates the handling of a service request in context of a positioning service. As soon as a new service request was received at the communication interface, the parent class type of the requested service is extracted from the request's data package. This information is necessary to check if another instance of the requested service class is already active. If so, the service quality provided by the already active service object is checked and compared to the requested minimum QoS (1). E.g., in context of a positioning system, the minimum QoS-parameters would be matched if the delivered positioning accuracy is the same or higher than the requested. Furthermore, also the update frequency for new positioning fixes should be equal or higher. If a service, which covers the demanded quality requirements is active, it can be used for replying to the request by registering a new callback and return it in a response package (2). Thus, the request was answered and the procedure terminates successfully. But sharing of services is not possible for all kinds of requests, e.g., a data transfer service cannot be shared due to individual parameters such as its destination address or the data to transfer. In this case a new private service object needs to be created and returned.

If there is no comparable service running, the system checks if the demanded service is a single request or if it requires periodic updates (3). As long as no periodic updates are required, an inquiry for cached data is directed to the Service Provider. Cached data is usable if a) it is output of the requested service class, b) it is of the demanded recency, and c) it is corresponding with the QoS-parameters described in the requests data package. If there is cached data of this kind (4), it can be packed into a response package and returned to the requester (5). In this case, the procedure terminates successfully.

Else, if the request was not of a single nature, or if there is no data of sufficient quality in the cache, the system checks for the

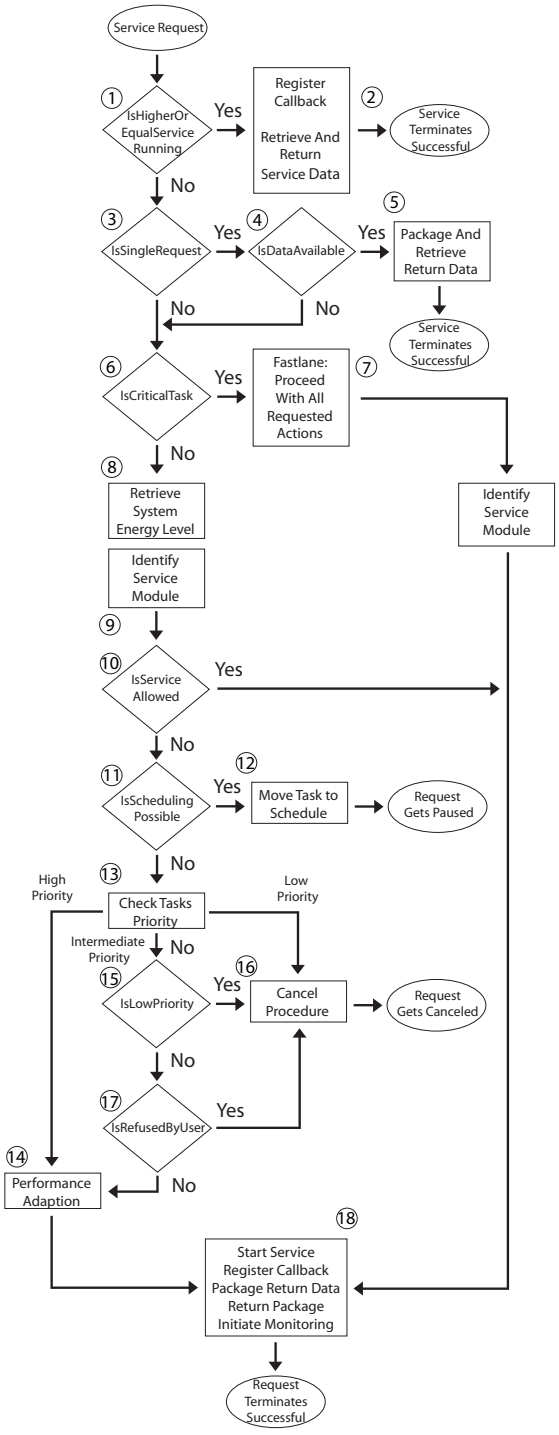

Figure 2. Handling of a service request with EMMA

request's priority in order to rate it critical or not (6) compared to other tasks. Critical tasks possess the highest priority in the EMMA system (see Section IV-G); their priority is rated critical by the user. Only special services, e.g., the device's emergency call function, are automatically classified as critical. If a request is rated critical, all demanded procedures are executed without further examination and disregarding any relevant system states (7), e.g., the current battery level. The response is returned by undertaking all needed actions (18) and the procedure terminates successful.

Failing this, a system check regarding its current energetic state is carried out (see Section IV-F), in order to discover potential resources applicable for new tasks (8). In favor of this goal, the energy consumption of currently active services is considered as well as predicted consumption of services which are scheduled for future usage. Additionally, information about the current battery level or the user's position (e.g., nearby power supplies) is also respected. That is why even with a drained battery the energetic system status of a device could be rated as non-critical, because the user is able to easily charge his phone.

If it turned out in step (8), that there is enough energy for the new service, the identification of a suitable service module is initiated (see Section IV-E). For this purpose, the Control Unit's Controller obtains a list containing all available service modules of the requested service class and forwards it to the Module Identificator. This unit tries to identify the best suited module by processing the class specific identification algorithm in combination with the QoS-parameters provided by the requester (9). In case that there is no module belonging to the desired service class, the procedure is terminated. If there are modules of the desired service class, but none of them matches the requested QoS in an exact way, a pointer to the module with most similar features and performance parameters is returned. In a worst case scenario, no module can be identified. In an unfavorable scenario, a module with a higher energy consumption than allowed or with a lower service quality than demanded is identified. That is why the performance and consumption parameters of a module again are verified and compared to their counterparts given by the system. Purpose of this check is to determine if the module's features are compatible to the system's energy state (10), before a new service object is instantiated combined with the integration of the identified service module. If a service creation is not possible, the system tries to schedule the task (12) without the need to cancel it. This procedure provides the feature of resuming to a task as soon as new resources become available again. But due to individual peculiarities, this is not possible for all services. Reasons for that may be a high service urgency, because of the service's individual features or due to a user's preferences; other services may only be reasonable at a given time. E.g., a navigation service is likely to be needed immediately, while a transfer service could possibly get scheduled for a future point in time. The decision of scheduling tasks or not is carried out by processing a given flag in the requester's package whether a task is schedulable or not. If not, the system continues with the verification of the request's priority (13) (see Section IV-G). The priority flag is also included in the request's data package and was extracted from the users preferences as well as by analyzing the devices usage history. After its extraction, the priority can be compared to priorities of other tasks. If the new task possesses the highest priority (14), compared to the tasks that are responsible

for scheduled or currently active services, the system tries to free energy for the new task by throttling, canceling or pausing active services. But the adjustment of active services needs to be handled with care to avoid possible inefficiencies. E.g., canceling a transfer service during a data transmission process could force the new transfer of the whole amount of data on resuming to a task. This would lead to a higher energy consumption compared to just finishing an ongoing transmission. If an upcoming task possesses the lowest priority level, the procedure is canceled immediately and the request gets marked as a failure (16). In other respects, if the task is prioritized with an intermediate level, the system again tries to adjust, pause or cancel services owned by lower prioritized tasks. If that is not possible or the additional provided energy is not sufficient, the user is invited to choose which services are important to him and which are likely to be canceled. In order to keep up the system's usability, this approach is only used in combination with higher prioritized tasks. Additionally, all taken adjustment actions are logged to enable a later on analysis of the systems behavior. In case that a service was approved in (10) or (7), or if enough energy could be freed after the prioritization procedure, a new service object, containing the identified module, gets created, followed by the preparation of a return package (18).

Generally, return packages can contain three different types of data: 1) a callback registered with a running service instance (2), 2) requested low level data sets provided by the cache (5) or 3) a freshly created private service object. After service usage, the service object is released by the requester and is not available for usage any more.

D. Communication interfaces

The EMMA concept defines a fixed set of internal and external interfaces used for communication. Internal interfaces cover up all routing information and services, e.g., to realize a working connection between a service object and its integrated service module (see Section V). In contrast, external interfaces are responsible for processing service requests and responding to them. Therefore, data bundles are used to maintain the communication between EMMA and any applications requesting a service. These bundles contain the ID of the requested service class, as well as information about the requester's service priority, the expected service runtime and QoS-parameters.

The structure of corresponding response bundles is similar. If a request is answered positively, the reply contains the requested service, as well as information about its actual instantiation, as the returned service may either be a callback function the requester can use, a dedicated service object or simply the requested data. The information about its actual instantiation facilitates correct data unmarshalling.

E. Identification of suitable services

In order to be able to choose the most appropriate service module, EMMA executes an individual identification algorithm for each service class. This is necessary to ensure that the performance of the chosen module matches both, the energy deallocated by the system and the demands of the service requester. Reasons for the necessity of having separate selection algorithms for each class are the different numbers of parameters, which are necessary for an adequate description of a service's energy consumption and its QoS-characteristics.

The simplest case of describing a module's characteristics is by the usage of two different performance parameters, which form a performance set. One parameter always represents the module's energy consumption, the other one describes a corresponding QoS-specific feature. Each module can possess any number of performance sets describing varying pairs of consumption and service quality related to different modes of operation. In this two dimensional space a target area can be identified between the point-of-origin and the intersection of the values of deallocated energy and requested service quality. In the following, the identification algorithm iterates across all existing modules of the requested service class and checks if any set of their performance parameters is located within the target area. If only one module can be found, this module will be selected. If several modules match the given thresholds, the one with the least energy consumption is selected. Otherwise, if no module can be found, the target area is extended and the same procedure is started again.

The complexity of the individual identification algorithms depends on the number of features, which are necessary for describing a module's performance. According to the current concept state, the mere provision of a service is regarded more important than the meeting of given performance parameters or the approved energy consumption. In case that no comparable module is offering the same service at a lower energy consumption, even a module with a much higher consumption than approved can be provided in order to meet the user's demands.

F. Continuous monitoring

In order to adapt the system's performance and to keep critical services running as long as possible, EMMA continuously monitors the system's energy state. Therefore, not only the current energy consumption but also the predicted consumption for future tasks, which can be determined by analyzing the active and upcoming tasks' schedules, combined with prediction techniques (see Section II-B3), is included. Additionally, relevant context information is considered, e.g., loading stations near a user's position (see Section IV-H). Each monitoring cycle is triggered by new incoming low level information concerning the system's power supply, e.g., the most recent battery level or changes in charging state. In order to visualize the monitoring process, Figure 3 shows each individual step which needs to be undertaken to control and adapt the system's resource usage. First, the system's current energetic state is determined by the Control Unit (1). The Schedule Manager provides information about effectively active services and the energy consumed by them. Together with predictions concerning the amount of energy that is needed for these and other upcoming tasks in the future, it can be forecasted if the energy will last for these duties (2). If there are no worries about the upcoming energy supply

and if there is no need - or not enough resources left - to upgrade the service quality of currently active tasks, the monitoring procedure terminates without further actions (5). Else, measures need to be initiated to adjust the system's energy balance (4). Generally, the system's performance can be adjusted into two directions: Given that either more energy than expected is available and the system's performance has been throttled before or a request demands a higher service quality than currently granted, the performance of a service as well as its QoS can be increased. In contrast, if there is less energy available than expected or a new high-priority service is started, the performance of active services can be throttled in order to free new resources. After undertaking demanded performance adaptations, the monitoring process jumps back to step (1) and the system's energetic state is checked again. If no further adaptations are necessary, the process terminates in (4).

To adapt the performance of a service module, it can be modified within its possible performance spectrum, e.g., throttle some or all of its individual performance attributes. If there are more adaptations needed, e.g., because of a better service quality or if even more energy is needed than the adjustment of a specific module can provide, the whole module can be replaced by another of the same service class but with different performance parameters. The adaptation of service performance commonly results in a change of service quality.

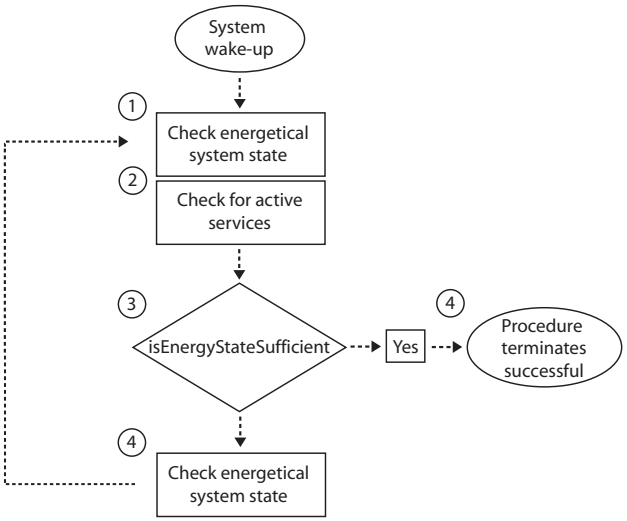

Figure 3. General overview of the EMMA concept and its components.

G. Task prioritization

Prioritizing tasks is one of EMMA's core concepts and not replacable for two different reasons. On the one hand, in order to make decisions concerning performance adaptations, a comparison as well as an evaluation of the scheduled service's performances is needed (see Section IV-F). On the other hand, there are services and applications with a critical priority, which means their execution has to be enforced despite a lack of energy or other resources.

Furthermore, as long as no user specific preferences concerning a service's priority exist, each service gets assigned with the standard priority `PRIORITY_MEDIUM`. In order to take account of the user's personal preferences, EMMA provides a granular priority concept. The priority levels, featured by the prototypical implementation are `PRIORITY_CRITICAL`, `PRIORITY_HIGH` and `PRIORITY_LOW`. Afterwards, they can easily be extended with custom priority levels. The declaration of a service as critical is meant to ensure its execution, even under unfavorable circumstances.

Moreover, the usage of priority tags enables the constitution of hierarchical layers between all existing services, applications and tasks, which results in an order for sequential task processing. The processing direction itself is determined by the current use case. E.g., if new energy resources need to be freed, services from the lowest hierarchical layer are adjusted, paused or canceled in the first instance. Else, if there is an existing energy surplus that allows a performance lift of before throttled services, services in the highest hierarchical layer are considered first.

H. Energetical relevant context

The usage of context with energetic relevance is also one of EMMA's key features. It is especially needed for making decisions concerning the usage of the mobile device's resources. The Energy Manager component, which is part of the Control Unit, is responsible for that.

Concrete statements concerning the system's current energy state can be determined by appropriate Application Programming Interfaces (APIs), which are part of every recent smartphone operation system (e.g., the Android BatteryManager [26] or the iOS UIDevice-object [27]). In contrast, the prediction of forecasts concerning upcoming tasks and related resource drain is much more difficult. To achieve this, logs about application or service usage can be used as well as information about calendar events, the user's social environment or related preferences (see Section II-B3). If it is obvious that there are not enough resources available for future tasks, the QoS can be adjusted. Alternatively, if a context information indicates that at the time of an upcoming task a power supply is available, possible resource shortages could be ignored.

Anyways, predictions are not only used for making decisions towards resource management, they can also be used for periodical resource consumption monitoring. If it is obvious that enough resources for the lift of a service's performance quality are available, some services can be selected for a performance increase by using the before described prioritization system.

V. Implementation

In the following, some technical details concerning the general implementation of EMMA's prototype as well the construction of individual service objects are presented.

A. Basic parameters

In order to assess the feasibility of our approach, we developed a prototypical application of EMMA with the aid of the Google Android API (target version 18). In its current form, however, the prototype was not yet integrated as an exclusive service manager into the operating system. Instead, due to access and rights restrictions of the Android SDK it has been implemented as part of the application layer. Our goal in this first version was to evaluate the technical feasibility of EMMA's core concepts without analyzing any possible increase in energetic efficiency in detail, yet.

In order to cope with EMMA's claim for modularity, the logic of service modules is capsuled in `.dex`-archive files, whose content is described both machine- and human-readable in corresponding `XML` manifest documents. The latter contain all information about a module's performance, nature, and energy consumption and can hence be used to dynamically identify any modules matching a service request. Our prototype is able to respond to incoming requests for either a continuous or a one-time positioning service by selecting the most appropriate service module, taking into account the current energy state of the device, as well as context information and currently active, or scheduled services.

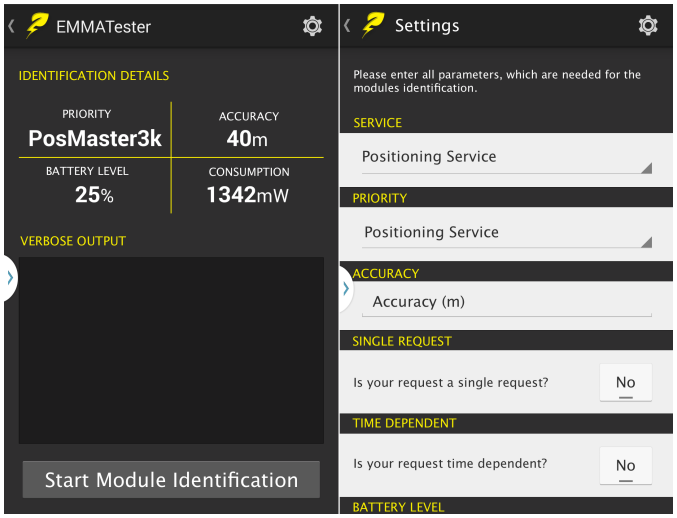

Figure 4. Screens of the EMMA prototype application.

To achieve this, the device's current energy consumption is determined at first. In order to keep the implementation overhead for the energy measurements as low as possible, the energy consumption is approximated by a simple model based on test results obtained from a HTC Desire using the PowerTutor application [8]. By combining these information with knowledge about active and scheduled services as well es their estimated runtime, it is now possible to calculate the energy remaining for new services. In order to identify a suitable service module, the requested QoS parameter contained in a request bundle and the amount of remaining energy are considered for building a target area (see Section IV-E). After identification, the chosen module is integrated into a singleton service object. According to the current concept, the identified module is not restricted in its energy consumption once it was installed, but it can be replaced by a more economical one if its consumption is extensive.

In order to be able to use the module's functionalities after its integration, a standard callback object provided by the Android SDK is installed in the service object, serialized and returned to the service requester. The latter is now able to listen for positioning updates after rebuilding the callback from its serialized version.

B. Constructing service objects

From a technical point of view, a service object consists of several individual components, which are visualized in Figure 5. In case of our prototypical implementation, each individual service object is derived from a basic service object and expands it with individual functionalities. Generally, a service object can be seen as an empty hull not realizing any kind of own logic. It merely implements all predetermined integration and communication gateways, in order to ensure a smooth integration of external logic modules. For the most a basic service object encapsulates all available information related to the service class it represents as well as the service modules which are allowed to become integrated. Moreover, it provides a standardized interface and related functions for accessing the service logic inside of the integrated module. On the upper left, all registered callbacks, which are organized in a list are shown. If a service is suitable for simultaneous usage by multiple users, a new user callback is registered here. This approach prevents the system from an unnecessary, parallel execution of multiple instances of the same service. Additionally, each registered callback contains a unique ID, which is returned to a service requestor after its registration. The ID ensures, that a callback can be identified and removed after service usage.

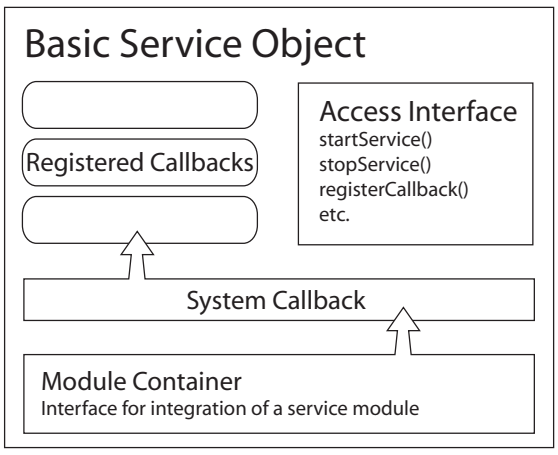

Figure 5. Schematic representation of a service object.

Furthermore, each service object possesses an individual and private system callback, which manages the internal communication to the service module. It enables the reception of

messages directly from the service module and mirrors them in a second step to each registered user callback.

On the upper right side of Figure 5, examples for possible access procedures are shown. These are necessary for a direct service or logic access, e.g., for delivering data sets or for starting or stopping a service, etc. All required methods for accessing a service class are defined in a unique way, which needs to be implemented by each module of this specific service class. Additionally, the module's container implements some further gateways for integrating the service module into its service object. At runtime, all logic contained in the module archive is loaded dynamically into the program code by using the Android-specific, reflection-based DexFile [28] and DexClassLoader-classes [29].

VI. Conclusion and future work

By introducing EMMA, we provided a holistic, modular concept for individual and context sensitive energy management, which meets the requirements stated in Section III. The modular concept applies facilities for user-centric service provision and renders the usage of partial optimization concepts as well as of their energetic savings potential. Furthermore, the provided architecture makes the integration of third-party logic a more trivial task for users and developers.

EMMA in its current state is to be considered a basic architectural framework. Lots of details need an initial clarification or further composition, e.g., the specification of different service classes and service modules. The process of identifying and providing services needs further improvements, as well. In its current state, tasks need a manual prioritization undertaken by the user. In order to automate this process, the usage of training or machine learning algorithms is conceivable. Furthermore, a capable security concept must be provided due to the possibility of including third-party code dynamically into the system at runtime. This feature indicates that additional security measures are inevitable in order to avoid the execution of malware within EMMA and to avoid the system's abuse.

In context of a following expert's discussion consisting of 8 participants with IT background, which was aimed to evaluate EMMA's feasibility and utility, we found that the procedure of integrating modules and the conventions for developing them, as well as the interface descriptions for requesting services need to become more facilitated. In their current state, they seemed to be not as clear as needed for an unrestricted and easy system usage. In a final step, a second implementation of the EMMA concept with the goal of evaluating its actual energy savings potential is to be carried out. Especially, the integration of the architecture as part of the operational system proves to be a challenge, since current platforms like iOS or Android do not allow this without significant interventions in their system's core.

References

[1] A. Ebert, F. Dorfmeister, M. Maier, and C. Linnhoff-Popien, "Emma: A context-aware middleware for energy management on mobile devices," in *CENTRIC 2014, The Seventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, pp. 48–53, IARIA, 2014.

[2] A. K. Dey, *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000.

[3] P. A. Zandbergen, "Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning," *Transactions in GIS*, vol. 13, no. s1, pp. 5–25, 2009.

[4] B. Li, J. Salter, A. G. Dempster, and C. Rizos, "Indoor positioning techniques based on wireless lan," in *LAN, First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, Citeseer, 2006.

[5] M. Schirmer and H. Höpfner, "Senst*: approaches for reducing the energy consumption of smartphone-based context recognition," in *Modeling and Using Context*, pp. 250–263, Springer, 2011.

[6] J. Manweiler and R. Roy Choudhury, "Avoiding the rush hours: Wifi energy management via traffic isolation," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 253–266, ACM, 2011.

[7] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM european conference on Computer Systems*, pp. 29–42, ACM, 2012.

[8] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 105–114, ACM, 2010.

[9] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, pp. 317–328, ACM, 2012.

[10] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive gps-based positioning for smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 299–314, ACM, 2010.

[11] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 315–330, ACM, 2010.

[12] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "Enloc: Energy-efficient localization for mobile phones," in *INFOCOM 2009, IEEE*, pp. 2716–2720, IEEE, 2009.

[13] F. Ben Abdesslem, A. Phillips, and T. Henderson, "Less is more: energy-efficient mobile sensing with senseless," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pp. 61–62, ACM, 2009.

[14] B. Priyantha, D. Lymberopoulos, and J. Liu, "Littlerock: Enabling energy-efficient continuous sensing on mobile phones," *Pervasive Computing, IEEE*, vol. 10, no. 2, pp. 12–15, 2011.

[15] M. Schirmer and H. Höpfner, "Senst*: approaches for reducing the energy consumption of smartphone-based context recognition," in *Modeling and Using Context*, pp. 250–263, Springer, 2011.

[16] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *Proceedings of the 9th ACM conference on embedded networked sensor systems*, pp. 82–95, ACM, 2011.

[17] N. B. Priyantha, *The cricket indoor location system*. PhD thesis, Massachusetts Institute of Technology, 2005.

[18] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking*, pp. 261–272, ACM, 2009.

[19] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "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, pp. 280–293, ACM, 2009.

[20] G. P. Perrucci, F. H. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the impact of 2g and 3g network usage for mobile phones' battery life," in *Wireless Conference, 2009. EW 2009. European*, pp. 255–259, IEEE, 2009.

[21] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 281–287, ACM, 2010.

[22] N. Ravi, J. Scott, L. Han, and L. Iftode, "Context-aware battery management for mobile phones," in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pp. 224–233, IEEE, 2008.

[23] E. Oliver and S. Keshav, "Data driven smartphone energy level prediction," *University of Waterloo Technical Report*, 2010.

[24] E. Oliver, "Diversity in smartphone energy consumption," in *Proceedings of the 2010 ACM workshop on Wireless of the students, by the students, for the students*, pp. 25–28, ACM, 2010.

[25] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz, "Primary user behavior in cellular networks and implications for dynamic spectrum access," *Communications Magazine, IEEE*, vol. 47, no. 3, pp. 88–95, 2009.

[26] "Android Developers - Monitoring the Battery Level and Charging State." http://developer.android.com/training/monitoring-device-state/battery-monitoring.html, 2015. [Online; accessed 06-January-2015].

[27] "Apple Developers - UIDevice Object." https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIDevice_Class/index.html, 2015. [Online; accessed 06-January-2015].

[28] "DexFile - Android Developers." http://developer.android.com/reference/dalvik/system/DexFile.html, 2014. [Online; accessed 25-July-2014].

[29] "DexClassLoader - Android Developers." http://developer.android.com/reference/dalvik/system/DexClassLoader.html, 2014. [Online; accessed 25-July-2014].