Noname manuscript No.
(will be inserted by the editor)

Transparent resource sharing framework for Internet services on handheld devices

Wouter Haerick · Femke Ongenae · Chris Develder · Filip De Turck · Bart Dhoedt

Received: date / Accepted: date

Abstract Handheld devices have limited processing power and a short battery lifetime. As a result, computational intensive applications can not run appropriately or cause the device to run out-of-battery too early. Additionally, Internet-based service providers targeting these mobile devices lack information to estimate the remaining battery autonomy and have no view on the availability of idle resources in the neighborhood of the handheld device. In this paper, we propose a transparent resource sharing framework that enables service providers to delegate (a part of) a client application from a handheld device to idle resources in the LAN the device is connected to. The key component is the Resource Sharing service, hosted on the LAN gateway, which can be queried by Internet-based service providers. The service includes a battery model to predict the remaining battery lifetime. We describe the concept of Resource-Sharing-as-a-Service that allows users of handheld devices to subscribe to the Resource Sharing service. In a proof-of-concept, we evaluate the delay to offload a client application to an idle computer and study the impact on battery autonomy as a function of the CPU cycles that can be offloaded.

Keywords Resource sharing · battery autonomy prediction · Internet services · handheld devices

1 Introduction

The widespread adoption of mobile, personal devices introduces challenges for Internet-based service providers. These personal, handheld devices have limited processing power and typically suffer from short battery lifetimes. As a result, network-intensive and CPU-intensive gaming or multimedia applications do not run smoothly and cause the device to run out-of-battery early. Additionally, Internet-based service providers

W. Haerick
Gaston Crommenlaan 8 bus 208, 9050 Ghent, BELGIUM
Tel.: +32 9 331 4940
Fax: +32 9 331 4899
E-mail: wouter.haerick@intec.ugent.be

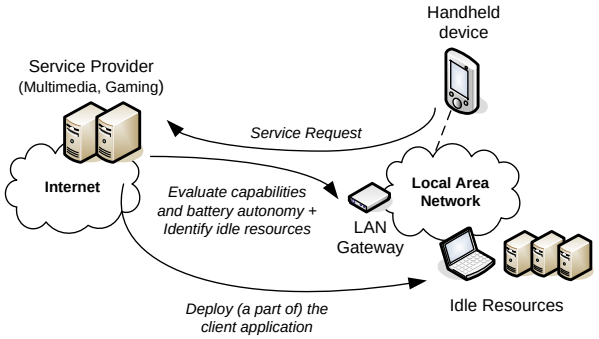

Fig. 1 Offloading (a part of) a client application from a handheld device to idle LAN resources

targeting the constrained, handheld devices experience a lack of information about battery autonomy and available CPU-power. No standardized technologies or solutions exist to query battery information of an end-user device, which hinders service providers to anticipate on low-battery or low-CPU situations. In this paper we will therefore present and evaluate a resource sharing framework that gives service providers a view on battery autonomy using the standards TR-069 [1]and UPnP [2]. In case a handheld device is connected to a LAN, surrounded computational power can be used to support a resource intensive application. Therefore at least a part of that client application should be offloaded from the handheld device to idle, surrounded computers. In figure 1, we illustrate a typical scenario where a handheld device initiates a service request towards a service provider. Upon this request, the service provider evaluates the capabilities, battery status and presence of idle resource by querying the LAN gateway, which is accessible from the Internet. In order to save battery and to relieve the CPU, two conditions need to be met. Firstly, sufficient idle resources should be available in the LAN. In [3] we have argued that the average CPU usage of desktop PCs in home and office networks is merely 12.9%. This leaves an important amount of idle CPU time that could be shared with CPU-constrained devices. Secondly, offloading CPU cycli from a handheld device towards a desktop PC should result in significant energy savings. This requires that the energy consumption of wireless data exchange with an idle computer should be less than the energy savings of the CPU. While power saving techniques exist to optimize energy consumption of the display and the network interface card, it is the aim of the Resource Sharing Framework to achieve similar power savings for the CPU without the need to dynamically alter the clock speed.

This paper is structured as following: In the next section we describe different strategies to relieve handheld devices. We refer to related work on offloading software components and optimizing battery autonomy, and describe three classes of scenarios that can benefit from resource sharing. In section 3, we recapitulate the shortcomings of related work and propose the software components of the Resource Sharing framework. We also describe the concept of Resource-Sharing-as-a-Service that allows users of handheld devices to benefit from resource sharing by subscribing to an Internet-based resource sharing provider. Subsequently, we present the battery model that is used by the Resource Sharing framework to estimate remaining battery lifetimes. In section 5, we evaluate the startup delay to offload a (part of a) client application and study

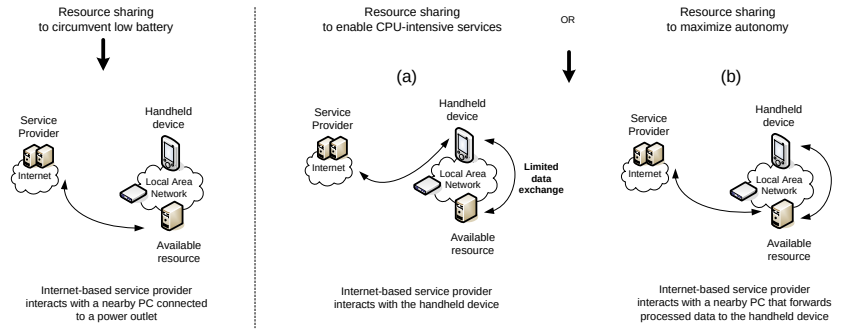

Fig. 2 Three classes of scenarios that benefit from resource sharing

the energy savings in case a local task is offloaded to a nearby, idle computer. These energy savings are studied as a function of the offloaded CPU. Finally, we conclude the benefits of the Resource Sharing framework in comparison with related work.

2 Strategies to relieve a handheld device

2.1 Three classes of scenarios

Different motivations may exist to relieve handheld devices and to consider offloading tasks to more powerful resources. As depicted in figure 2, we have identified three classes of scenarios that benefit from resource sharing:

- *Resource sharing to circumvent low-battery issues*: In case a handheld device is in a low battery mode, or in case a service provider may predict that the handheld device will encounter battery issues during service execution, a resource sharing framework allows to transfer the client application to another computer. As a result, an user of a mobile device can initiate a service from the handheld computer. However, the service provider will establish a session with a nearby, idle LAN resource.
- *Resource sharing to enable CPU-intensive services*: If the handheld device does not meet the minimal CPU requirements of a service, offloading a part of the client application to an idle resource has the advantage that the user may continue to use his personal device for the user input, however complex calculations can be delegated to a more powerful resource in the LAN.
- *Resource sharing to maximize battery autonomy*: In a third class of scenarios, where a handheld device has sufficient battery and meets the CPU requirements, the use of a resource sharing framework might be beneficial to optimize the battery autonomy. In section 5, we will evaluate the conditions that need to be met to increase the battery autonomy.

In the first class of scenarios, the service provider will establish a service session with the idle, powerful LAN PC. Given that the exchange of data between the handheld device and the LAN resource consumes a considerable amount of energy (consumed by the wireless network interface card and the CPU), it is not energy efficient if the handheld device forwards broadband data streams towards the idle resource. Therefore,

in case broadband data streams need to be preprocessed by the idle LAN PC, the session with the service provider should be set-up with the LAN PC and not with the handheld device. This is what is illustrated on the right in figure 2.

2.2 Related work

Our work mainly relates to two aspects of resource optimization: (1) optimal offloading of software components in a distributed environment and (2) adopting strategies to optimize the energy consumption of mobile devices.

Offloaded software components Considerable research has been performed with respect to offloading computations using software agents [5–7]. However, whereas these studies strive to improved service execution times and better overall resource utilization, our RSF distinguishes itself by minimizing the energy consumption of local resource utilization on battery-powered devices by means of battery autonomy prediction. In [6, 7] the run-to-completion times have been optimized, together with the overall processor idle time, by coupling predictive application data with scheduling heuristics. The authors of [7] propose an iterative heuristic algorithm to obtain load balancing across multiple processing nodes. In [5] an energy-aware middleware is proposed to share resources across home networks using virtualization technology. Given the small topology of LAN networks and its mostly constant network delays, the presented scheduling algorithm are not applicable to the Resource Sharing Framework we propose. Additionally, these scheduling algorithms and middleware solutions do not consider battery autonomy of constrained devices.

Optimizing energy efficiency Energy efficiency is an important topic in the area of ubiquitous computing and has been discussed for many years. Different strategies have been proposed to allow constrained devices to spend more time in a low(er)-energy state. Amongst these energy reducing techniques are display power saving modes [9, 10], dynamic voltage scaling of CMOS based processors [11, 12], network protocol optimizations reducing the number of (re-)transmissions and collisions [13, 14], and transmission power control mechanisms [15, 16]. In the case of PDAs, the maximum energy consumption of the wireless 802.11 interface is significantly higher than the maximum for the display or CPU. Reducing the transmission rate of data will however have negligible impact on the per-packet energy consumption as not the bitrate but the active time is a measure for energy consumption [17, 18]. As a result main energy optimization should be realized by shaping the idle time of a wireless interface considering different usage patterns. In this paper we explore the potential, local energy savings of offloading a part of the application from a mobile, constrained device to a nearby personal computer.

3 Transparent Resource Sharing Framework

In this section we propose the main components of our transparent Resource Sharing Framework (RSF). The framework overcomes the shortcoming of grid computing solutions by focusing on battery autonomy prediction instead of load balancing and optimized scheduling. Additionally, our resource sharing framework is based on the

standards TR-069 and UPnP, respectively for remote management and LAN-side service discovery and execution. With a remote management interface, our framework allows for the integration with any service provider that has established a trust relation with a trusted third party managing the LAN network.

3.1 Requirements for adoption by broad range of devices

The involved actors in our Resource Sharing Framework are (1) the handheld device, (2) the service provider, (3) the LAN gateway and (4) a pool of idle, powerful resources. The handheld device and the idle resources need to be monitored in terms of battery status and available CPU-time. We assume that the LAN gateway collects all these monitored data, and provides an interface to trusted Internet-based providers. In order to allow adoption of the RSF by a broad range of devices, we have identified the following requirements:

- Device compatibility and portability: Partly as result of the rapid adoption of wireless technologies, LAN networks are evolving towards highly heterogeneous networks. The RSF should therefore be able to cope with devices running different operating systems and varying hardware. Operating system specific CPU and battery monitoring calls should therefore be clearly separated from the service oriented resource sharing calls. To increase portability, the RSF is implemented as a java service platform with modular OSGi services. Automatic detection of the operating system allows integrating with the proper native libraries for battery and CPU monitoring.
- Standards-based: The RSF should avoid imposing additional standards to LAN networks. It was therefore a design decision to adopt not only the discovery mechanism of UPnP [2], but also to specify each capability in the RSF as an extended basic UPnP device. While the emerging UPnP-protocol is used to describe the capabilities, support for TR-069 [1](and its extensions) is foreseen for remote management purposes.

3.2 Framework components

The RSF is implemented as a java-based, modular software framework with the core intelligence deployed in the LAN gateway, compliant with the architectural gateway architecture presented in [4]. Each module complies with the OSGi specifications [19] and can be remotely deployed into the OSGI service platform. The monitoring component, to monitor CPU and battery, is also implemented as an OSGi module and as a result requires the Java-based OSGi service platform on each of the involved handheld and idle devices.

Figure 3 gives an overview of all the Resource Sharing Framework modules distributed across the involved devices. The following service components need to be deployed in the LAN network to enable resource sharing scenarios:

- TR-069 Resource Manager (TR-069): The TR-069 Resource Manager, hosted by the LAN gateway, enables an Internet-based provider to remotely allocate idle resources by sending SOAP messages that comply with the TR-069 standard.

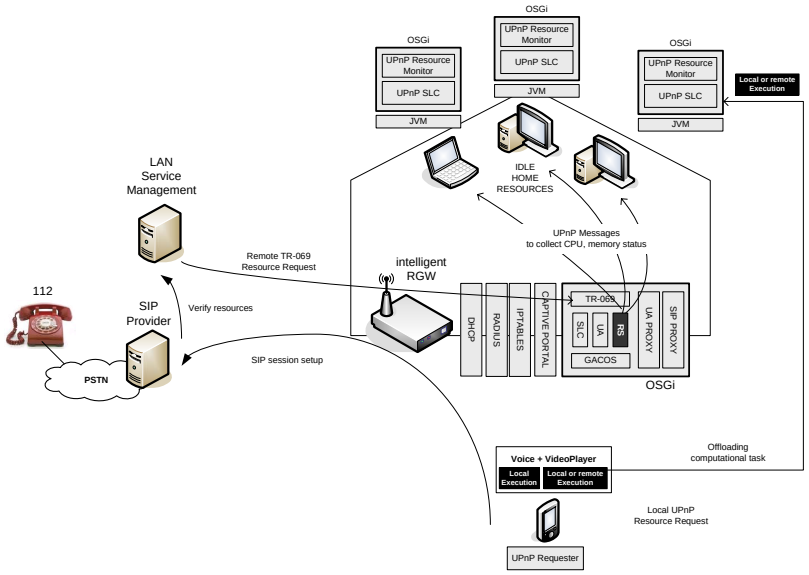

Fig. 3 Software components of the Resource Sharing Framework deployed on the handheld device, idle computers and LAN gateway

- UPnP Resource Sharing (RS) Service: This UPnP service, hosted by the LAN gateway, takes the actual resource sharing decisions based on the monitored data, admission criteria, job priorities and resource allocation strategy.
- UPnP Resource Monitors: Each participating home computer runs a UPnP Monitor service that can be queried to retrieve information about CPU, memory, storage, battery and the java virtual machine. This UPnP service communicates with the ResourceWrapper service that handles the native calls to the underlying operating system.
- UPnP Resource Requesters: Constrained devices may run a UPnP Resource Requester service. A local service that can distribute computational intensive tasks, can use this UPnP control point to find any available resources inside the home.
- UPnP SLC Service: Each participating LAN computer also runs an UPnP ServiceLifeCycle (SLC) Service that allows to install and run computational tasks in favor of handheld devices

3.3 Resource-Sharing-As-A-Service (RSaaS)

The presented RSF supports the concept of Resource-Sharing-as-a-Service (RSaaS) and as a result enables LAN Service Management providers to offer resource sharing as a value-adding service. The role of LAN Service Management provider could be played by the Internet access provider, which already manages the Internet connectivity to the LAN gateway. Figure 4 depicts the mobile user that subscribes to RSaaS at a LAN Service Management provider. Upon acceptance of the subscription request, the LAN Service Management provider is able remotely deploy the presented RSF components as

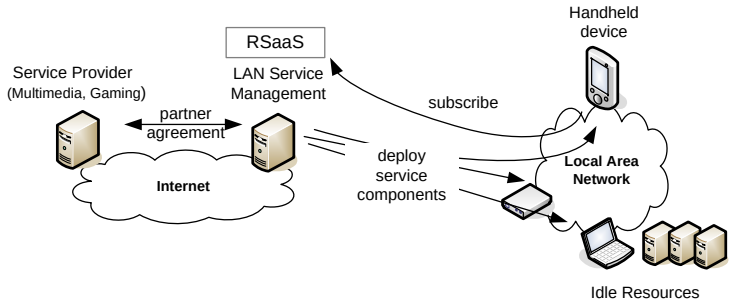

Fig. 4 Resource-Sharing-as-a-Service offered by a LAN Service Management Provider

these are OSGi modules that can be installed using a standard TR-069 install message. This install message triggers the Service Lifecycle Component of the LAN gateway. As a consequence, all components can be remotely deployed in a way which is transparent for the end-user. The requirements for these automated installations are a TR-069 enabled LAN gateway, and an OSGi service platform installed on all participating LAN devices. Service providers that support resource sharing, which means that their client software can be (partly) offloaded to a computational computer, need to have an agreement with the LAN Service Management providers in order to let mobile users benefit from their RSaaS subscription. For each service request, that originates from a LAN network that has subscribed to RSaaS, the Internet-based service provider should include the LAN Service Management provider to identify idle resources to optimize battery autonomy and to enable resource-intensive services that would otherwise not run. For security reasons, the LAN gateway should only process resource sharing request originating from a trusted LAN Service Management provider. The TR-069 specification recommends the use of SSL to guarantee the authenticity of the requestor of TR-069 messages.

Referring to figure 3, it is illustrated how a SIP provider may involve a LAN Service Management provider to transfer the video transcoding part of a SIP video session from a handheld device to a computational resource elsewhere in the LAN network. It is the LAN Service Management provider who communicates with the TR-069 Resource Sharing Manager on the LAN gateway. This TR-069 service interacts with the core Resource Sharing (RS) service. The latter makes a decision to transfer the transcoding service to an idle resource in order to extend the battery lifetime of the handheld device.

4 Battery autonomy prediction

The core Resource Sharing service, installed on the LAN gateway, uses a battery autonomy model we proposed in previous work to predict the remaining autonomy of handheld devices. This battery model assumes four main energy consumers: display, CPU, wireless network interface card and I/O operations. As the energy consumption depends on the type of hardware used, a calibration step is required to define the device-specific model parameters. Table 1 below shows the model parameters we defined for the iPAQ hx2490 performing the tests as described in [3].

Table 1 Parameter values of the battery autonomy model for the iPAQ hx 2490 for the display, CPU and wireless interface card

Model parameter	Setting	Energy (mWh)
$E_{display}$	50% brightness	922 mWh
E_{CPU}	520 MFLOPS	842 mWh
$E_{CPU_{data-out}}$	100KB	0.035 mWh
$E_{NIC_{data-out}}$	Sending	1250 mWh
$E_{CPU_{data-in}}$	100KB	0.801 mWh
$E_{NIC_{data-in}}$	Listening	102 mWh

The battery model considers two energy penalties in case a part of a service is offloaded: (1) the additional energy required to run the RSF and (2) the additional energy to exchange data between the handheld device and the idle, computational resource. These two energy penalties should be compensated by the energy gain of the CPU. Figure 5 illustrates the potential energy savings of the iPAQ hx2490 in case a locally running application is partly offloaded to a powerful PC. To evaluate the battery autonomy as a function of the offloaded CPU, we introduce the leverage factor L:

$$L = \frac{CPU_{offloaded}}{CPU_{local}} \tag{1}$$

As the potential energy savings not only depend on the offloaded CPU, but also on the data exchanged with the computational resource, figure 5 shows the battery autonomy for different active times of the wireless network interface card (WNIC).

Figure 5 illustrates that for a locally running application on the iPAQ hx2490 with 50% of the CPU offloaded to a computational resource (this means L=1), energy can be saved if the WNIC is not longer than 25% of the time active. In case the WNIC is active for more than 50% of the time, no energy gains are possible. Within the RSF, this battery model is used to predict the remaining battery lifetime. The Resource Sharing service on the LAN gateway monitors the actual battery status, and using the CPU and network requirements provided by the service provider the remaining battery time

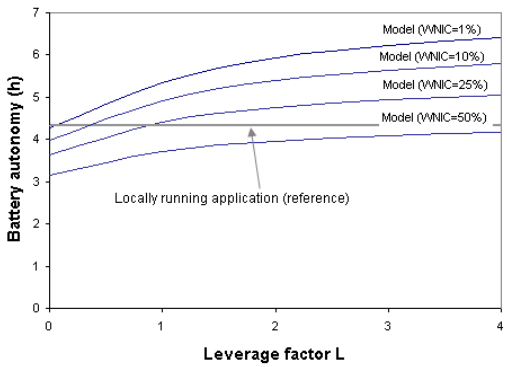

Fig. 5 Battery autonomy model for different active times of the wireless network interface card (WNIC)

Table 2 Breakdown of the time to offload the video service to an idle resource, and to release allocated resources as soon the video service has been stopped.

Execution step	Average delay (ms)
Resource allocation	1299
Install offloaded service	737
Start offloaded service	1010
	3046 (offloading)
Stop service	363
Uninstall service	792
Release resources	631
	1768 (releasing)

is calculated. If the remaining battery time is less than the expected service execution time, the Resource Sharing service tries to allocate an idle resource to transfer the complete service from the handheld device to a nearby computer.

5 Performance evaluation

5.1 Evaluation of startup delay

We have implemented a proof-of-concept of the Resource Sharing Framework with an iPAQ hx2490 running the UPnP Resource Sharing Monitor on top of a J9 IBM JVM, one desktop PC acting as LAN gateway and two other PCs acting as idle, computational resource. The PDA runs a local OSGi service platform with an OSGi video player. Instead of playing the video from the local disk, the PDA requests an idle resource inside the home network to run a video transcoding bundle. This transcoding bundle is able to stream the movie with an average bitrate of 86.6 Kbps, compared to the local movie with an average bitrate of 941 Kbps. For this specific configuration, however, the graphical processing of both scenarios loaded the CPU for 100% and, as a result no energy gains could be obtained by playing the remote, transcoded movie. The demo setup is however useful to evaluate the time delay introduced by the resource sharing framework. Table 2 illustrates the time split to start an offloaded service as well as the time split to de-allocate resources.

The 1299 ms for resource allocation is mainly spent in the DomoWare UPnP basedriver which is responsible for the translation between the XML-formatted UPnP messages and the java objects. In total it takes 3 additional seconds to start an application that requires remote execution. To optimize this delay, the dependent UPnP base driver of DomoWare and the default OSGi installation functionality of the Knopflerfish software need to be analyzed.

5.2 Evaluation of battery autonomy

The same setup is used to illustrate resource sharing for CPU intensive services, and to validate the energy model we proposed in [3]. For this test setup, an OSGi service generates a CPU load of 100% on a handheld device by calculating the value of Pi with a certain accuracy. This computational task takes 941 seconds on the hx2490 PDA and

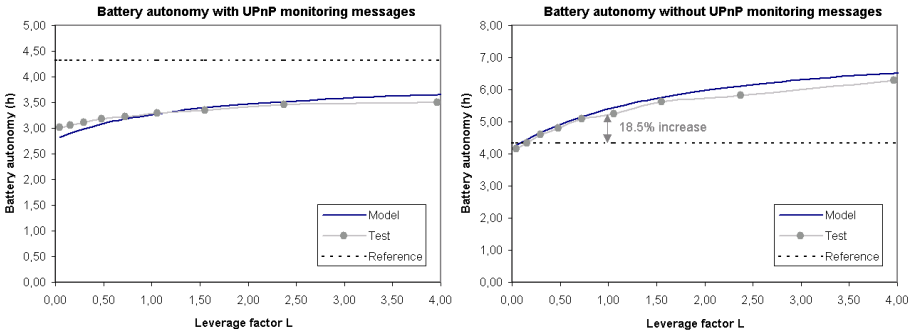

Fig. 6 Evaluation of battery autonomy model for a CPU-intensive application service that is partly offloaded to an idle computer. The graph on the right shows the battery autonomy in case the UPnP resource information updates are turned off, whereas on the left the handheld device regularly sends UPnP messages with resource information.

only 2 seconds on a 2,8 GHz pentium desktop PC. Consequently, resource sharing not only saves CPU power, it can also result in reduced calculation times and hence higher responsiveness. The demo service has been designed as two interacting software components: One local parent component that manages temporary results, and one child component performing the calculations. The latter can be offloaded to an idle resource. In this setup, the handheld device discovers itself an idle resource in its neighborhood and deploys the child component on that idle resource. By varying the number of decimals to be calculated locally on the handheld device - in favor of more decimals calculated remotely - energy consumption is measured for different leverage factors. In order to request a nearby resource at the Resource Sharing service and to transfer state data between the two application components, 13KBytes of outbound data and 11KBytes of inbound data are processed by the WNIC of the PDA. This requires less than a second of send time. Figure 6 depicts the battery autonomy calculated with the energy model, compared to the autonomy derived from the measured energy consumption. Although an energy gain was expected as the result of very limited send time of the WNIC, we observe a high energy consumption that fits with the energy model for WNICactive = 72%. Two effects are causing this loss of energy: (1) Every 5 seconds the PDA sends its resource information to the UPnP Resource Sharing Service and hence wakes up the WNIC, (2) the WNIC of the hx2490 remains in a high power state for several seconds after sending has completed.

Turning of the UPnP resource information updates sent from PDA to the RGW results in the expected energy gains as illustrated on the right of figure 6. For a leverage factor L=1, meaning that 50% of the local computations are offloaded, this figure illustrates that resource sharing delivers an increase in autonomy of 18,5%.

6 Conclusions

The widespread adoption of handheld devices poses challenges for Internet-based service providers, in particular for resource intensive gaming and multimedia services. For handheld devices that are connected to a LAN, offloading a part of the service to an

idle LAN resource may considerable increase the autonomy of the handheld device. In contrast to resource scheduling algorithms for large-scale grid computing architectures, which focus on the overall optimization of resource utilization, we presented in this paper a Resource Sharing Framework (RSF) that optimizes battery autonomy. The core Resource Sharing service uses a battery model to predict the autonomy and to decide for complete or partial offloading. As the RSF is based on the well-adopted standards UPnP and TR-069, it allows Internet-based service providers to remotely offload a part of a service from a mobile, battery-powered devices to an idle LAN resource. We discussed potential energy gains for the iPAQ hx2490 and illustrated quantitative energy savings that highly depend on the active time of the wireless network interface card. We described how the RSF supports the concept of Resource Sharing as a service. As a result, resource sharing could be offered as a novel, value-adding service by Internet access providers.

Acknowledgements C. Develder is supported by the Research Foundation - Flanders (FWO–Vl.) as a postdoctoral fellow.

References

1. DSLHome Technical Workgroup, TR-069: CPE WAN Management Protocol, http://www.broadband-forum.org/ (2004).
2. Universal Plug and Play Forum, http://www.upnp.org/.
3. Haerick, W., De Winter, D., Vandenberghe, P., De Truck, F., Dhoedt, B., Acke, W., Breaking the barriers of constrained, mobile devices in the home, *In the proceedings of the Asian International Mobile Computing Conference 2007*, p35-44, Calcutta, India (2007).
4. C. Wu, C. Liao, and L. Fu, Service-oriented smart home architecture based on osgi and mobile agent technology, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, pp. 193205 (2007).
5. Hlavacs, H., Weidlich R., Hummel K., Houyou A, Berl A., de Meer H., Distributed energy efficiency in future home environments, *Annales des telecommunications*, vol. 61, 9-10, pp. 473-485 (2008).
6. Abraham, A., Buyya, R., Nath, B., Nature's Heuristics for Scheduling Jobs on Computational Grids, *In International Conference on Advanced Computing and Communications* (2000).
7. Cao, J., Spooner, D., Jarvis, S., Saini, S., Nudd, G., Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling, *In International Parallel and Distributed Processing Symposium* (April 2003).
8. Nudd, G., Kerbyson, D., Papaefstathiou, E., Perry, S., Harper, J., Wilcox D., PACE : A Toolset for the Performance Prediction of Parallel and Distributed Systems., *Int. J. of High Performance Computing Applications, Special Issues on Performance Modelling*, 14(3), pp. 228-251 (2000).
9. Margi, B., Obraczka, K. Manduchi, R., Characterizing System Level Energy Consumption in Mobile Computing Platforms, *In IEEE WirelessCom 2005 - Symposium on Mobile Computing* (2005).
10. Choi, I., Shim, H., Chang, N. Low-power color TFT LCD display for hand-held embedded systems, *In ISLPED* (2002).
11. Gruian, F., Hard real-time scheduling for low energy using stochastic data and DVS processors, *In Proc. of Intl. Symp. on Low-Power Electronics and Design* (Aug. 2001).
12. Grunwald, D., Levis, P., Farkas, K., Morrey, C., Neufeld, M., Policies for dynamic clock scheduling, *In Proc. of 4th Symposium on Operating System Design and Implementation* (Oct. 2000).
13. Chen, J., Sivalingam, K., and Agrawal, P., and Kishore, S., Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption, *IEEE Infocom'98*, San Francisco, USA, pp. 150-157 (March 1998).

14. Gauthier, P., Harada, D., Stemm, M., Reducing Power Consumption for the Next Generation of PDAs: It's in the Network Interface!, *Proceedings of MoMuC '96* (September 1996).
15. Lee, D., Panigrahi, D., Dey, S., Network-aware image data shaping for low-latency and energy-efficient data services over the palm wireless network, *In WWC (3G Wireless)* (2003).
16. Kravets R., Krishnan, P., Power Management Techniques for Mobile Communication, *In MOBICOM Conference Proceedings* (1998).
17. Lattanzi, E., Acquaviva, A., Bogliolo A.,. Run-time software monitor of the power consumption of wireless network interface cards, *In Proceedings PATMOS-04*, pp. 352-361. SpringerVerlag (2004).
18. Xu, R., Li, Z., Wang, C., Ni, P., Impact of data compression on energyconsumption of wireless-networked handheld devices, *In Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems*, IEEE Computer Society Press, Los Alamitos, CA (2004).
19. OSGi Alliance, OSGi Server Platform Release 4 (Oct. 2005).