SANDMAN: an Energy-Efficient Middleware for Pervasive Computing

Gregor Schiele
Information Systems II
Universität Mannheim
Schloss, 68131 Mannheim, Germany
gregor.schiele@uni-mannheim.de

Christian Becker
Information Systems II
Universität Mannheim
Schloss, 68131 Mannheim, Germany
christian.becker@uni-mannheim.de

ABSTRACT

Energy efficiency in pervasive computing is crucial for devices operated by battery. To provide energy efficiency we created an energy efficient middleware, called SANDMAN. In this paper we present an overview on the past research done in the SANDMAN project and the current and future directions of our work.

1. INTRODUCTION

Energy is a crucial resource in pervasive computing systems with mobile devices. These devices are often embedded into everyday items and can not be provided with a large battery or a fixed connection to the power grid. Thus, the efficient operation of devices with respect to energy is a major challenge of such systems. When designing our pervasive computing middleware BASE [2], we experienced this challenge and decided to integrate algorithms and mechanisms for energy-efficiency in our middleware.

When starting our work, we looked at the main sources of energy consumption. The first thing we learned is that while a lot of work has been done to lower the energy consumption for sending and receiving data, a large additional amount of energy is consumed by idle devices waiting to be used. As an example, it takes 805 mW to keep an IEEE 802.11 network interface up and running without sending data [3]. Idle devices provide currently unneeded and thus unnecessary resources and consume energy by doing so. This energy can be saved by temporarily switching such devices in a low-energy sleep mode. However, doing so results in a number of challenges that must be addressed to keep the system operational, e.g., network connectivity and service discovery.

In this paper we report on the challenges we met when designing an energy-efficient middleware for pervasive computing that allows to power down currently unused devices. We also discuss solutions to overcome these challenges and present the current and future work done in the project.

The paper is structured as follows. First, we define our system model and assumptions. After that we describe the features of our existing middleware BASE that are needed to understand our approach. Following to this we present our approach towards an energy-efficient middleware and evaluate our middleware briefly. Finally, we discuss current work and how we plan to proceed from the current project state, give the related work, and finish the paper with a short conclusion.

2. SYSTEM MODEL AND ASSUMPTIONS

Our targeted system class consists of a number of battery-operated mobile devices. Each device is equipped with one or more network interfaces. Using these interfaces the devices form a number of wireless mobile ad hoc networks (MANETs). Basic MANET functionality, e.g., addressing, remote execution, is offered by our pre-existing middleware, which we assume to be installed on each device. In addition, we assume that each device has two operational modes: a fully operational AWAKE mode and an energy-efficient SLEEP mode. While in SLEEP mode, the device can not perform any operations or communicate. To switch back to AWAKE mode, the device has an internal timer, e.g., a watch dog timer, which reactivates the device after a predetermined time. This simple model can be extended to multiple different operational modes. However, in this paper we restrict ourselves to the simple model to ease the description of the main concepts used in our approach.

3. BASE

We designed our energy-efficient middleware as a number of extensions to our existing middleware BASE. This allowed us to concentrate on the new challenges imposed by our need to save energy while reusing a lot of previous work. Before describing our extensions in more detail, in this section we present the parts of the basic middleware that are needed to understand the extensions.

BASE is designed to be a minimal yet flexible communication middleware for pervasive computing. It does not rely on any external infrastructure, enabling the devices to cooperate with each other in a peer-to-peer fashion.

The architecture of BASE is shown in Figure 1. The middleware is structured as an extensible micro broker. The broker itself manages interactions with remote devices and synchronizes them with respect to the application's desired interaction model. To communicate, (semantic, serializer, modifier and transceiver) plugins are used to add support for different communication technologies and protocols. As an example, to access a CORBA service, the device developer only has to integrate an IIOP plugin into the BASE configuration. The management of these plugins is the responsibility of the plugin manager. At runtime the middleware detects nearby devices (with a discovery plugin), negotiates communication abilities with them and allows the local application to access other devices using a service abstraction. Once an interaction takes place, BASE automatically builds a suitable protocol stack by selecting and integrating multiple plugins. To adapt to networking changes, BASE is able

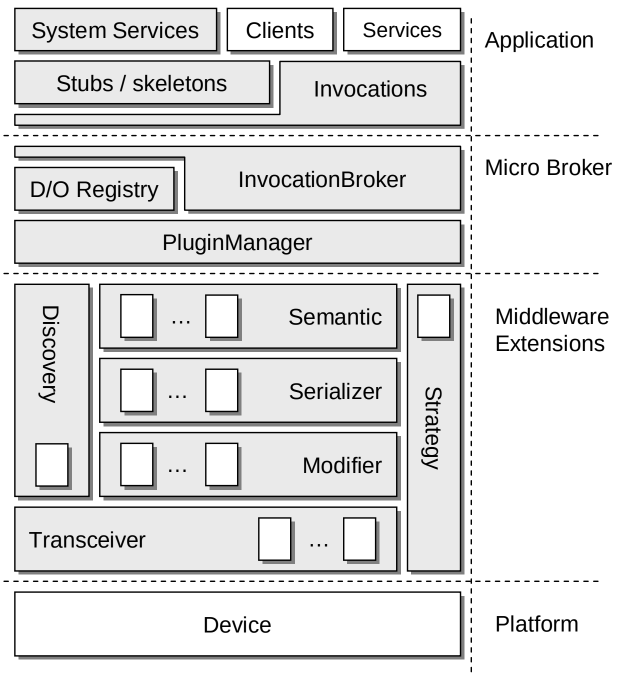

Figure 1: BASE Architecture

to reselect the used plugins dynamically. More information about BASE can be found in [2].

4. SANDMAN

To add energy efficiency support to our existing middleware BASE, we extended it with a number of system services to allow an energy-efficient operation of each device. The resulting new middleware system is called SANDMAN. It stays fully compatible with existing BASE installations. SANDMAN is designed around three main concepts to save energy:

1. Reduce the energy consumed by transferring data by selecting the most energy-efficient communication protocols that are available in a given situation.

2. Switch idle devices to their low power SLEEP mode to reduce unnecessary standby energy consumption.

3. Allow clients to select the most energy efficient service to create energy-efficient application configurations.

The first concept can be realized easily with BASE using its existing ability to select plugins dynamically. To do so, the plugin descriptions must be enhanced with information about their energy consumption and a suitable selection strategy must be provided and integrated. Whenever a new protocol stack is selected, the selection strategy accesses the plugin descriptions and selects the most energy efficient configuration. The second concept, the deactivation of idle devices presented us with a number of challenges, which we discuss in the following sections. The third concept, the selection of energy-efficient services is subject to current and future work and discussed later.

4.1 Transition Scheduling

The first challenge when deactivating idle devices is to schedule deactivations properly. Often, it is not easy to decide whether a device is unused and can be deactivated. It may be idle at the moment but play a crucial role in the execution of an application in the near future. As an initial approach, we relied on a transition strategy with a fixed inactivitiy threshold. Such approaches are well known from the area of Dynamic Power Management. They can be implemented very efficiently even on resource-restricted devices, as they only require a timer to operate. In addition, we added an interface to the middleware that allows application code, e.g., service implementations, to explicitly specify that the device is currently in use and should not be deactivated. Further information is provided by the BASE microbroker, which notifies SANDMAN about incoming and pending requests, as well as currently used local services. This rather simple approach works well in cases where specific usage patterns are difficult to determine. In other scenarios, more complex idle detection mechanisms, e.g., based on statistical approaches, could be beneficial. Finally, BASE handles each interaction between a client and a service individually. While this results in a very flexible system, we decided to add an additional abstraction for service usages, so-called *sessions*. Using a session, a client can specify that it currently uses a given service. This information is then forwarded to SANDMAN which will not deactivate the device offering the service, even if there is no client interaction for some time. To cope with suddenly disappearing clients, leases are used. In addition, sessions can be used by clients to negotiate with the service that the latter may sleep even while the client is using it, e.g., because the client can cope with a given latency. Client and server can also negotiate synchronization times, i.e., they will communicate at given times only, allowing both to temporarily sleep. In our implementation, the ability to open a session is provided but negotiation strategies are subject to future work and must be provided by application developers at this time.

4.2 Service Discovery

Before using them, clients must first discover devices and their services. However, existing discovery approaches like UPnP or Jini cannot handle deactivated devices and wrongly assume that they have left the system. Thus, before deactivating a device, we must make sure, that it stays discoverable. To do so, we developed a self-adaptive discovery protocol that can handle deactivated devices. Our approach works as follows: at startup time, each device operates autonomously and answers discovery requests from remote clients directly. In this state the system resembles a classical UPnP discovery system. During the system operation, the devices cluster themselves with neighboring devices that have the same mobility pattern as themselves. This ensures that the resulting clusters are highly stable, which is necessary to achieve long sleep times without introducing errors in the discovery process. Otherwise, devices that left the communication range of their clusters while sleeping could lead to phantom discoveries. Each cluster has a single leader, the so-called cluster head (CH). Once a cluster is formed and a CH elected, all devices in the cluster switch their discovery system to a registry-based approach, resembling Jini. The CH collects information about all services in its cluster and answers discovery requests from clients for

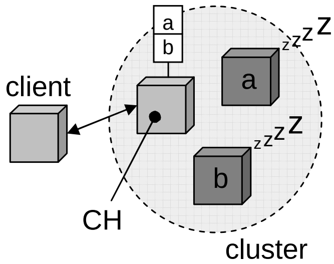

Figure 2: SANDMAN Approach

them as shown in Figure 2. This allows all other devices in the cluster to switch to their SLEEP mode, while the CH keeps advertising them. In addition to this, the CH can act as a proxy to detect new services for sleeping client devices. To accept new client requests or receive information about newly detected services, each device in a cluster awakes regularly.

An overview of the protocol used by SANDMAN to put devices to their SLEEP mode is given in Figure 3. In this example, we assume that a cluster consisting of two devices n_1 and n_2 has already been formed and omit the messages necessary for cluster management. At the beginning, n_2 starts its inactivity threshold timer. If n_2 is idle for t_is, it decides to go to sleep and sends a $SLEEP\_ANC$ message to its CH n_1, including the desired sleep time t_{sd}. The CH can modify this sleep time to allow cooperative scheduling algorithms as discussed later. It stores the new sleep time t_s in its local database for n_2 and sends back a $SLEEP\_ACK$ message with the sleep time. After receiving this message, n_2 configures its internal watch dog timer to reawake after t_ss and transitions to its sleep mode. Meanwhile, a client device n_3 contacts the CH to search for services. The CH finds that n_2 offers a service suitable for n_3 and announces this to the client device. In this message, it includes the service descriptions, the plugins that can be used to contact the device as well as the remaining sleep time of n_2 (zero if the device is awake). The client waits for the specified time until n_2 awakes. Then, it contacts n_2 directly and uses its service. A special case arises, if the device wants to sleep shortly after a client device discovered one of its services. The CH cannot know, if the client will contact the device and thus denies any sleep requests from a device, if the time between its last discovery and the sleep request is smaller than a given threshold t_a. Once a service is no longer used, its device restarts its inactivity threshold timer and the algorithm starts anew. More information about the service discovery approach and the protocols used (e.g., for clustering) can be found in [9] and [8].

4.3 Connectivity Preservation

In addition to keeping the devices discoverable, the network connectivity must be maintained. If we deactivate devices at will, we will most likely lower the connectivity of the network. We may even induce network partitioning. Luckily, we can reuse the solution chosen for the discovery and put the responsibility for routing on the CHs. In addition, we have to make sure that the CHs form a connected overlay network and can reach all nodes. To do so we design our clustering approach such that it not only uses the mobility patterns of devices for its clustering decision but also the current neighbor graph of the devices. Two devices are clustered iff they have the same neighbors. This makes sure that each one of them can act as CH and will be able to reach all neighbors. An example for this are devices carried by the same user. These devices are nearby and typically have the same neighbors. Note that to really ensure this property, we have to recheck it regularly to cope with later connectivity changes.

This approach consumes additional energy, first because fewer devices are clustered and second to perform the regular check. If we can accept a certain (small) loss in connectivity, we can schedule the rechecks to occur only rarely or omit them altogether. In addition we can accept a certain amount of difference in neighboring sets when clustering devices, e.g., we cluster devices when their neighborhoods overlap by at least 90%. Using these parameters we can adapt the system behaviour between more connectivity preserving and more energy-efficient as needed.

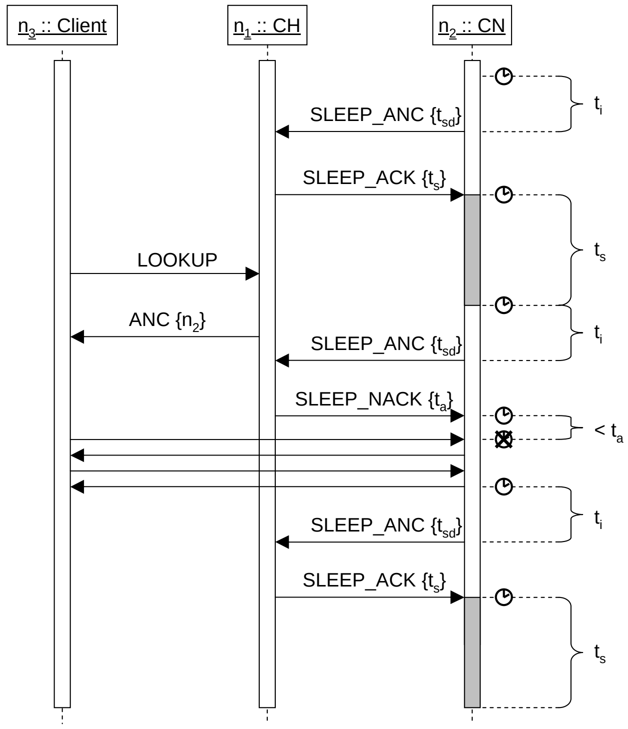

Figure 3: SANDMAN Protocol Overview

4.4 Interaction Latency

When a device is asleep it cannot be reactivated preliminarily, e.g., to handle an unexpected request by the user. Clearly, in some cases the user might be able to manually reactivate a device prior to its scheduled awake time by pressing a special button, etc. However, we do not assume that this is always possible or even the normal case. Thus, a client wanting to use a sleeping device must wait until the device awakes on its own. This slows down the client's execution and may consume additional energy. Therefore, it is important to lower the experienced interaction latency. To do so, we propose to cooperatively schedule the sleep times of all devices in a cluster. To realize this, SANDMAN allows

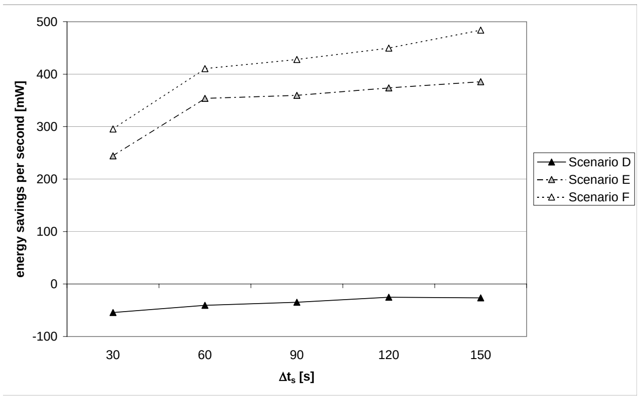

Figure 4: Energy Savings

CHs to manage the sleep schedule of its whole cluster locally and to coordinate all devices accordingly. Currently, we are examining two cooperative scheduling algorithms: the first interweaves the sleep times of devices offering the same service such that the time until one of these devices awakes is minimized. The second keeps one device awake all the time, allowing clients to use a service without any additional delay. The device that must stay awake is chosen by the CH in a round-robin fashion. Our current implementation includes only a simple scheduling algorithm that operates on isolated devices. Cooperative scheduling algorithms are subject to current and future work.

4.5 Evaluation

To evaluate the energy savings that can be achieved by putting idle devices into their SLEEP mode, we performed a number of experiments using the Network Emulation Toolkit (NET) [5]. NET is a Linux-based emulation environment developed at Stuttgart University for testing and evaluating network protocols in both stationary and mobile environments. For our experiments we defined scenarios with different mobility characteristics, e.g., device speed and device group size. Figure 4 shows the resulting energy savings for three scenarios with a device speed of 2 m/s and three different group sizes, single devices (Scenario D), groups of 4 (Scenario E) and groups of 10 devices (Scenario F). Clearly, a group size of one leads to the well known random waypoint model. The results are shown for different sleep times Δt_s and are averaged over all devices in a single cluster, i.e. they include the overhead experienced by the CH.

In Scenario D, the devices consume more energy than without SANDMAN. This is due to the fact that devices are clustered rarely and the message overhead due to clustering consumes more energy than is saved by sleeping devices. Therefore, for this scenario, SANDMAN is not beneficial and should not be used. However, for larger group sizes, the devices are able to save up to 484 mW per node for Δt_s=150 s and a group size of 10. For the chosen continuous device consumption of 805 mW, this is a saving of approximately 60% per device, including CHs and unclustered devices. For scenarios with other movement speeds the results are accordingly, while total values for higher speeds are lower. This is the case, as with higher mobility, clusters become less stable and devices must recluster more often. We can observe the same effect when comparing scenarios with identical group sizes but different movement speeds. The achieved energy savings are lower for higher speeds. A much more elaborate evaluation of our approach, including message overhead, savings, delays and discovery errors can be found in [8].

5. CURRENT AND FUTURE ACTIVITIES

SANDMAN so far provides basic functionalities to enable energy-efficient device operation. However, different possibilities to further enhance the achievable energy savings exist and are currently evaluated by us. The most promising ones are discussed in the following sections.

5.1 Transition Scheduling

The transition scheduling strategy currently implemented in SANDMAN does not take into account other devices in a cluster. Instead it operates completely isolated. In addition, we use fixed, preset values for the parameters involved in the strategy, e.g., the inactivity threshold and the chosen maximum sleep time. Clearly, there are a number of possibilities to enhance this approach. First, we can enhance the inactivity threshold strategy by using dynamic parametrization. Second, we are currently examining more advanced transition strategies, e.g., based on statistics, to provide us with better predictions of future device usages. Third, we already developed a first cooperative transition scheduling algorithm, which takes into account all services in a cluster when computing sleep times. This algorithm must be evaluated and analyzed further.

5.2 Service Selection

A major issue in service oriented systems is the selection of suitable services by clients. From an energy efficiency point of view, this selection should depend heavily on the resulting energy usage. Thus, if multiple services are available, the client should use the one which leads to the most energy efficient application configuration. However, without system support, the client cannot decide which one is this. The resulting energy consumption depends on many factors and cannot predetermined with total certainty. As some prominent examples, the energy consumption depends on the amount and frequency of communications between client and server, the local execution cost of the service on its device, and the additional consumption if the service uses additional services to provide its functionality. In addition, the stability of the resulting configuration must be taken into account. A service might be highly energy efficient but is expected to become unavailable in short time, leading to another application reconfiguration with additional costs.

In the future, we want to provide additional support for selecting energy-efficient services. To do so, we plan to develop additional algorithms to model and predict the resulting energy usage of different configurations. First, we can use an analytical model to compute an estimated consumption. Second, we can rely on historical data, i.e. measurements taken for past configurations (see, e.g., [6]). In reality, we expect solutions that combine these two approaches to provide the best trade off between complexity and energy-efficiency.

5.3 Session Negotiation Strategies

As described before, sessions allow clients and services to negotiate energy saving strategies, e.g. by specifying common synchronization points. Although SANDMAN already allows such negotiations, additional support to do so would be beneficial. Most importantly, different strategies must be developed and analyzed to help application developers to decide on the best strategy for their code.

5.4 Adaptive Service Discovery

The Consumptions for discovery and usage must be carefully weighted against each other. It may be beneficial (at least from the system's perspective) to use a slightly worse service that was discovered with much less effort. Again, the precondition to follow this approach is the provision of exact and efficient models to estimate the energy consumption of a future service usage. Once this information is available, the system can adapt its discovery efforts depending on the achievable savings. As an example, if an application uses a very energy consuming service at the moment, SANDMAN can increase the frequency and range of discovery requests to find a better service. On the other hand, if a nearby energy-efficient service is used, the need for additional discoveries is low and the middleware can decide to stop searching for alternative services, until the used one becomes unavailable.

6. RELATED WORK

There are a number of existing energy-efficient middleware systems complementing our approach. The GRACE project [7] aims at reducing the energy-consumption of mobile devices that process multimedia data. It combines system functions like process scheduling, CPU power management and data encoding to enable global adaptation. The MillyWatt project [10] enables battery-powered devices to run for a predefined period of time. To do so, active devices are deactivated periodically for a specific fraction of time. In contrast to this, we deactivate idle devices, only. The Power Aware Reconfigurable Middleware (PARM) [4] and the Remote Processing Framework (RPF) [6] enable energy savings on mobile battery powered devices by shifting energy intensive tasks to resource rich devices. Our approach is able to do this by modelling such tasks as services that can be executed remotely. However, we currently do not support clients in selecting whether a given service should be executed locally or remotely. Another approach is taken by MagnetOS [1]. Through the continuous redistribution of application parts across the available devices of a mobile ad hoc network, MagnetOS reduces the communication cost by reducing the length of data paths.

Regarding energy-efficient service discovery, the DEAP-Space system enables devices to safely deactivate their communication adapters. To keep devices discoverable, it uses synchronized time windows to broadcast service announcements in a single hop environment. Our approach is aimed at multi hop networks and does not require synchronized devices, enabling optimized interaction latencies.

7. CONCLUSION

In this paper, we have presented our energy-efficient middleware SANDMAN. SANDMAN is realized as a number of extensions to BASE, our minimal and adaptive communication middleware for peer-based pervasive computing environments. It supports energy-efficient communication by selecting energy-efficient protocol stacks, and deactivates idle devices to reduce the idle standby energy consumption. To do so, SANDMAN clusters devices dynamically depending on their mobility patterns and neighboring devices. This allows to deactivate devices while preserving the network connectivity and the discovery of the devices and their services. Work is going on in different directions. Most prominently, we expect energy efficient service selection to emerge as a major enhancement to save additional energy. To do so, the future energy consumption of different application configurations must be predicted, e.g. using suitable analytical models or historical measurements.

8. REFERENCES

[1] Barr, R., Bicket, J., Dantas, D., Du, B., Kim, T., Zhou, B., and Sirer, E. On the need for system-level support for ad hoc and sensor networks. *ACM SIGOPS Operating Systems Review 36*, 2 (April 2002).

[2] Becker, C., Schiele, G., Gubbels, H., and Rothermel, K. Base – a micro-broker-based middleware for pervasive computing. In *Proceedings of the IEEE international conference on Pervasive Computing and Communications (PerCom)* (Mar. 2003).

[3] Feeney, L. M., and Nilsson, M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)* (Anchorage, AK, USA, April 2001).

[4] Mohapatra, S., and Venkatasubramanian, N. PARM: Power aware reconfigurable middleware. In *Proceedings of the IEEE 23rd International Conference on Distributed Computing Systems (ICDCS)* (May 2003).

[5] NET - Network Emulation Testbed. Webpage. http://net.informatik.uni-stuttgart.de/.

[6] Rudenko, A., Reiher, P. L., Popek, G. J., and Kuenning, G. H. The remote processing framework for portable computer power saving. In *Proceedings of the ACM Symposium on Applied Computing (SAC99)* (Feb. 1999), pp. 365–372.

[7] Sachs, S., Yuan, W., Hughes, C., Harris, A., Adve, S., Jones, D., Kravets, R., and Nahrstedt, K. GRACE: A hierarchical adaptation framework for saving energy. Technical Report UIUCDCS-R-2004-2409, University of Illinois, February 2004.

[8] Schiele, G. *System Support for Spontaneous Pervasive Computing Environments.* PhD thesis, University of Stuttgart, 2007. Adviser-Kurt Rothermel and Adviser-Christian Becker.

[9] Schiele, G., Becker, C., and Rothermel, K. Energy-efficient cluster-based service discovery. In *Proceedings of the 11th ACM SIGOPS European Workshop* (September 2004).

[10] Zeng, H., Fan, X., Ellis, C., Lebeck, A., and Vahdat, A. ECOSystem: Managing energy as a first class operating system resource. In *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)* (Oct. 2002).