Resource Management for Particle-Computers

Tobias Zimmer, Frank Binder, Michael Beigl, Christian Decker and Albert Krohn
Telecooperation Office (TecO) University of Karlsruhe
Vincenz-Priessnitz-Strasse 1, 76131 Karlsruhe, Germany
http://www.teco.edu
{zimmer,binder,michael,cdecker,krohn}@teco.edu

ABSTRACT
We present a system for real time management of the resources of Particle-Computers. The particle-Computers are a type of Smart-Its - a Ubiquitous Computing platform equipped with sensing, computing and communication hardware. Our management system provides the developer with easy access to real time features needed in almost every application for Ubicomp environments that is based on periodic or sporadic evaluation of sensor values.

Keywords
Real-time, resource management, Particle-Computer, developer support

INTRODUCTION
Particle-Computers (Figure 1) are technically advanced Smart-Its [1], a Ubiquitous Computing platform that was developed in our lab at TecO under the roof of the Smart-Its project [2]. As most platforms for context aware computing, Particle-Computers feature a number of input channels including different sensors and inbound communication, a computation unit for analyzing and processing of contexts and output channels like actuators and outbound communication.

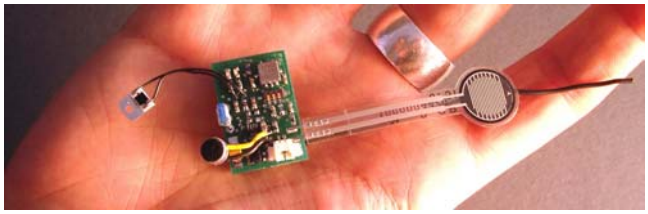

Figure 1: Particle-Computer

Many applications in Ubiquitous Computing involve data gathering or the provision of newly generated context information in per defined periodic time intervals as well as sporadic when changes in the environment are detected. These parallel functions like sampling different sensors, computing new contexts and communicating can best be implemented in separate tasks. Thereby it is more important to be able to guarantee a maximum time for an operation to be completed, like taking sample form a sensor, than just to complete every operation as fast as possible. So we developed the P-RMS (Particle Resource Management System) to provide real time scheduling functionality of the resources of Particle-Computers to the software developer. The system is intended to manage the execution of multiple (real-time) tasks with a minimal overhead on our Ubiquitous Computing platform.

Software Architecture
For providing maximum performance given the limited computing power and the small amount of available memory, the software of the P-RMS was split in two main components: a runtime environment, implemented on the Particle-Computer platform and a development tool running on a standard personal computer.

P-RMS DEVELOPMENT TOOL
The P-RMS development tool takes over some of the functionality of a real-time resource management system that can be applied at development time of the software for Particle-Computers. This is reasonable due to the resulting reduction of the load on the Particle-Computers at runtime. Functions that where transferred to a powerful personal computer are the feasibility computation of a given set of tasks, the check of the reservation of shared resources other than the processor and the generation of a runtime configuration for the Particle-Computer program.

To achieve maximum flexibility it is possible to feed the development tool with a configuration containing different real-time task-sets that may be executed on the Particle-Computer alternatively. Thus we overcome the disadvantage of a single predefined task-set at development time. For all task-sets the feasibility computation is done separately. This allows us to switch between the different sets of tasks at run time on the Particle Computers.

The scheduling we perform for the task-sets is an earliest deadline first (EDF), non-preemptive scheduling strategy without inserting idle times and using dynamic priority for tasks. Multiple sets of real-time tasks and one background task can be scheduled. The system supports temporal as well as permanent resource reservations. Schedulability computation is performed for non-concrete task-sets containing sporadic and periodic tasks according to the formulas in Zeng and Shin [3], that were adapted to our special requirements.

P-RMS RUNTIME ENVIRONMENT
The runtime environment of the P-RMS includes the scheduler, a real-time clock (RTC) and management routines for switching between task-sets and single tasks. It needs about 5,5 Kbytes of program memory; the exact amount of data

memory required depends on the number of tasks in all task-sets and the maximum number of instances of theses tasks. It can be computed to

$(96 + MaxNumberOfInstances * 4 + NumbeOfTasks * 4)$ bytes.

This is feasible as the Particle-Computers are equipped with 32 Kbytes of program memory and 1536 bytes of data memory, leaving enough recourses for user applications, e.g. a typical test configuration we used contains 4 real-time tasks and a background task needs about 132 bytes of data memory.

The P-RMS runtime environment provides various functionalities to the applications running on the Particle-Computers. This includes the management of periodic tasks by setting the period length and ensuring that they are started periodically. Furthermore, the runtime environment creates sporadic tasks based on input events and sets their starting time. The runtime environment also includes the service routines for switching between the different predefined task-sets.

Scheduler

The scheduler is responsible for assigning the processor and the other allocated system resources to activated tasks in the order of their priority and running the background task when the processor is not assigned to a real-time task. Priorities of the real-time tasks are assigned following the EDF scheduling strategy. The P-RMS scheduler works very efficient due to the fact that the schedulability tests are performed at development time of the application software. This guarantees that only schedulable task-sets are contained in any given application.

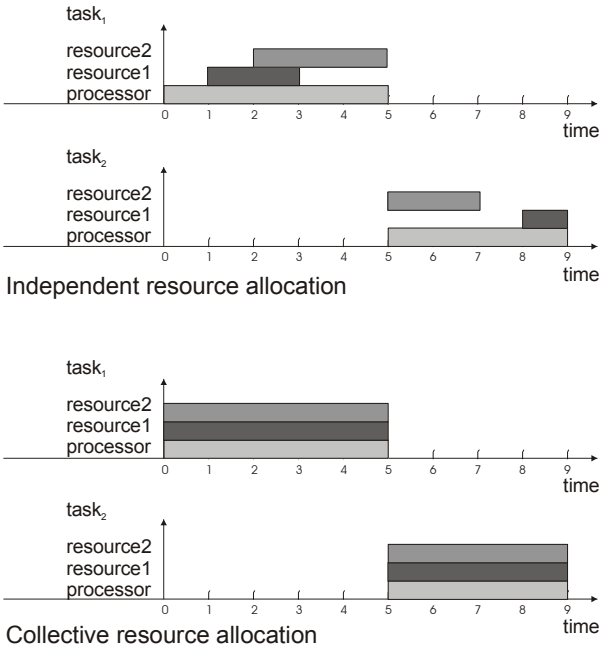

Figure 2: Assignment of resources performing independent or collective allocation

Resource assignment in general can be performed independent for each available resource or collective for all resources allocated by one task (see Figure 2). The advantage of independent resource assignment is that any given resource is allocated only as long as it is needed. This enables maximum parallelism of tasks. The disadvantage of that approach is, the schedulability of every resource has to be checked separately and dependencies between reservations have to be handled explicitly. In the P-RMS we decided to go for collective resource assignment due to the fact that only one processing unit is available and no virtual parallelism of tasks can be introduced performing non-preemptive scheduling. Details on all design decisions in P-RMS can be found in [4]

IMPLEMENTATION AND TESTING

The implementation of the P-RMS followed the "test first" strategy, known from extreme programming [5]. Using this method, tests for the functionality of every unit of code are designed and implemented prior to the implementation of the code unit itself. This results in an early detection of errors in the implementation.

EVALUATION AND FUTURE WORK

The evaluation of the P-RMS is still in process. We were able to determine some areas where further improvements in the performance and memory consumption of the system may be possible. E.g. one major improvement will be a further reduction of the runtime of the scheduler on the Particle-Computers. The maximum runtime of the scheduler depends on the maximum number of instances of tasks in a task-set. This maximum is seldom reached, so performance enhancements can be achieved by better prediction of those maxima. Additionally, a simplification of the RTC structure could reduce the runtime of a RTC-query from 299 cycles to 26 cycles at the expense of some loss in comfort in reading the current time and data. Another improvement we already identified for future implementation is the introduction of a hierarchical ordering of the resources. This will simplify the reservation of compound resources.

REFERENCES

1. Michael Beigl, Tobias Zimmer, Albert Krohn, Christian Decker, and Philip Robinson. *Smart-its - communication and sensing technology for ubicomp environments*. Technical Report ISSN 1432-7864 2003/2, April 2003.
2. The Smart-Its Project. http://smart-its.teco.edu. 2003.
3. Q. Zeng and K. G. Shin. *On the ability of establishing real-time channels in point-to-point packed-switched networks*. IEEE Transactions on Communications, vol. 42(2/3/4) :1096-1105, February/March/April 1994.
4. Frank Binder. *Ressourcenverwaltungssystem für Particle-Computer*. Master thesis at the TecO, University of Karlsruhe. May 2003.
5. Ron Jeffries, Ann Anderson and Chet Hendrickson. *Extreme Programming Installed*. Addison Wesley. ISBN 0-201-708-426 October 2000.