Real-Time in Plan 9: a short Overview

Pierre G. Jansen
University of Twente
PB 217, 7500 AE Enschede
Netherlands
jansen@cs.utwente.nl

Sape Mullender
Lucent Technologies
Bell Laboratories Murray Hill, NJ 07974
United States
sape@plan9.bell-labs.com

April 14, 2003

Abstract

When *shared resources* are involved, scheduling in current hard real-time operating systems, too often has its timely behaviour *guaranteed* at the cost of a rather complicated administration. We will show that we can improve this considerably by using methods based on so-called *Real-Time Transactions* (RTTs). A RTT is a task that has guaranteed the use of all needed resources after it has started, without ever having to wait for the resources' release; an RTT is only started if these resources are free. RTTs allow for a complete separation of a *real-time application* and its involved *system support.* Scheduling, (shared) resource synchronisation and admission control is executed automatically by the underlying system, while an application programmer only needs to specify timing constraints (deadline, period, runtime) and resource needs. We will discuss the implementation of RTTs within Plan 9 as used at Bell-labs, and we will illustrate the straightforward and elegant use of our transaction scheduling theory. [1].

Keywords: transactions, inheritance, schedulability, run-to-completion semantics

1 Introduction

Embedded systems have become an important part of all the ubiquitous computers found in our every day environment. These systems span a wide variety of different tasks, from simple to very complex; they may house appliances for control such as in our cars, communication for hand-helds and support for multimedia, all requiring some form of timely behaviour. Embedded system are controlled by computer systems running general purpose operating systems when possible or real-time operating systems when needed.

In this article we describe the metamorphosis of the general purpose operating system Plan 9 [6] to a real-time operating system. Although other operating systems may also have real-time support, we believe there are only few general purpose operating systems with a comparable native support for real-time (RT) applications.

[1]This research is sponsored by the IBM Equinox Grant.

Plan 9 is relatively little known and has but a small user community (a few thousand installations). Nevertheless, it is a complete operating system and it is the only operating system booted by many of its users. Plan 9 is also used in several embedded environments.

The background theory we use for the real-time metamorphosis of Plan 9 can be used for any other operating system. In this article we present basic strategies for scheduling real-time tasks running on a single processor. This theory is based on so called transactions as introduced in [4]. Transactions present a simple model for RT activity and they will be explained in detail. A transaction's main characteristic is that it may only start whenever all resources it needs during runtime are free. Consequently a started transaction never has to wait for a resource in use. An important consequence is that transitive waiting and hence deadlocks will be avoided. The use of transactions makes scheduling theory comprehensible and simplifies the basic scheduling protocols and feasibility analyses to elegant and light-weight algorithms. Transactions can be used to extend *Rate Monotonic* (RM), *Deadline Monotonic* (DM) [1], *Stack Resource* (SR) [2], and/or *Earliest Deadline First* (EDF) [5] protocols.

In the current context we confine ourselves to an adapted EDF protocol that can handle the use of shared resources on basis of static *deadline inheritance*, similar to the static priority inheritance used in the Priority Ceiling (PC) protocol [7]. We call our adapted version the EDF Inheritance (EDFI) protocol. EDFI allows for the a simple specification on application level of a RT task. Feasibility analyses of the task set and admission control can be done automatically. Scheduling and synchronisation of task is completely handled at system level. EDFI has been chosen to enhance the real-time version of Plan 9.

Transactions hold their needed resources during the course of their activity (including preemption). This can prevent preemption of higher priority tasks if during the course of activity the resources are only needed part time. Part time resources are only used during a predefined amount of time which must be specified on beforehand by the application. If part time resources are claimed and released in a nested way, equal to the nested critical sections in other protocols (like the Priority Ceiling protocol) , then EDFI holds its attractive properties. In the subsequent sections, we shall describe our system and present main parts of the theory behind it.

2 Conclusion

The real time scheduler is installed in the currently distributed version of Plan 9 (obtainable through `plan9.bell-labs.com`). It has already been used in several applications, one of them an experimental wireless base station.

Discussions about whether or not to include support for resource sharing in our real-time scheduler was won by the resource-sharing camp when the algorithms presented here emerged: the schedulability test is not overly complicated and the run-time complexity is practically O(1): only the queue insertions are not constant-time operations, but the queues are invariably very short. In addition, the scheduler prevents resource contention from causing gratuitous context switches and it is completely deadlock free. Finally, the same scheduler can trivially be used for preemptive or non-preemptive real-time EDF scheduling.

One real-time application we built has nothing but shared resources: the Clockwise [3] mixed-media file system has many real-time processes with varying periods and costs sharing disks. As it turned out, scheduling the disks was much more important than scheduling the CPU. The disk scheduling and its feasibility analyses could be handled with exactly the same algorithms as those for processor scheduling by treating the disk as a single resource, shared by every distinguished task that uses it.

The combination of EDFI and Plan 9 has shown to be a successful one. We believe that EDFI should be used one more platforms.

References

[1] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard Real-Time Scheduling: The Deadline Monotonic Approach. In *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software*, Atalanta, 1991.

[2] T. P. Baker. A stack-based resource allocation policy for realtime. In *Proceedings: Real-Time Systems Symposium*, pages 191–200. IEEE Computer Society Press, 1990.

[3] P. Bosch, S. J. Mullender, and P. G. Jansen. Clockwise: A Mixed-Media file system. In *IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS)*, volume II, pages 277–281, Firenze, Italy, Jun 1999. IEEE Computer Society Press, Los Alamitos, California. `http:// www.cwi.nl/ ~peterb/ papers/ icmcs99.ps.gz`.

[4] P. G. Jansen and R. Laan. Scheduling techniques and quality of service of real-time kernels. Technical report TR-CTIT-98-05, Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, Feb 1998.

[5] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[6] Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom. Plan 9 from Bell Labs. *Computing Systems*, 8(3):221–254, Summer 1995.

[7] L. Sha, R. Rajkumar, S. H. Son, and C. H. Chang. A real-time locking protocol. *IEEE Transactions on computers*, 40(7), July 1991.