

Classes of Algorithms for Predicting Timing Behavior in Embedded Systems

Prof. Dr. Horst Wettstein, Vector Informatik GmbH

Introduction

In many systems embedded in a technical context it is essential that certain timing conditions are fulfilled. That means that activities can be guaranteed to produce their effects at times that are needed to maintain an orderly processing. In such cases it may be useful to have a means to predict the behaviour of the system before it is executed. More precisely, one declares deadlines for a collection of periodic activities and applies a test stating whether all the activities meet their deadline or not (schedulability). Many algorithms for such testing have been developed and successfully applied. For some time now it has been observed that such methods increasingly penetrate the area of automotive information processing. In this whitepaper **Vector Informatik** presents a short survey of the most common types of algorithms for predicting especially the CAN bus behaviour.

Survey of Algorithms

Surveying the literature one comes across three classes of algorithms for testing the schedulability of queuing systems. Using the most characteristic feature as an identification we may call them

- Limiting Value Test
- Initial Sector Test
- Tabular Test

In this order the cost of calculation increases. However, also the applicability becomes broader and the effectiveness more powerful. In the following we will sketch some of these methods. In most cases the tests assume a preemptive scheduling policy. However, this kind of behaviour is not always given. This already indicates the fact that the tests may inherently lead to approximations (actually bounding to worst case behaviour).

Limiting Value Test

One of the oldest and simplest test is based on a limit value theorem [LL73]. The test assumes that the activities are ordered according to their period (rate monotonic analysis), that they are scheduled preemptively, and that their deadlines are equal to their periods, a rather specialized application. The theorem states, that n activities can meet their deadlines, if for the processor load L the formula

$$L \leq n * \left(2^{\frac{1}{n}} - 1 \right) \quad (1)$$

holds. The condition is to be interpreted as an implication, that is if it does not hold the schedulability can be given or not, no statement is possible. Fig.1 sketches the formula. For instance if two activities with period time 10 and 16 and computation time 5 and 4 respectively, the load adds to $L = 5/10 + 4/16 = 0.75$ which is less than $2 * (\sqrt{2} - 1) = 0.828\dots$. The activities may safely be executed. The limiting value for $n \rightarrow \infty$ approaches $\ln(2) = 0.693\dots$.

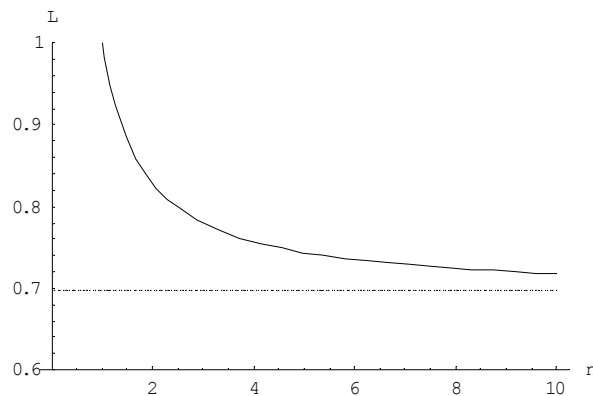


Fig. 1. Limiting load value versus number of activities

Initial Sector Test

A more general test offers the Initial Sector Analysis which was also presented in [LL73]. We again assume periodic activities a_i $i \in [1, n]$ with period P_i , computation time C_i , and deadline $D_i \leq P_i$. Liu and Layland were able to prove, that if periodic activities with arbitrary deadlines less than or equal to the period time are ordered according to their priorities (mostly corresponding to rate monotonic or earliest deadline monotonic order) and all activities are ready to be processed at the same time (this feature modelling the worst case coincidence), then it computationally suffices to show, that each activity meets its deadline once. That is only an initial part of the time axis has to be considered. The basic idea with respect to an activity a_x expresses the fact that within the response time $R_x \leq D_x$ of a_x all the repetitions of higher priority activities fitting into R_x have totally to be executed in addition to the computation time C_x of a_x itself. Cast into a formula this reads as follows:

$$R_x = C_x + \sum_{i=1}^{x-1} C_i * \left\lceil \frac{R_x}{P_i} \right\rceil \leq D_x \quad (2)$$

The ceiling brackets count for the fact that only integral parts of computation times may be squeezed into R_x . Since R_x appears on both sides, on the right side even in a non extractable way, the formula can not explicitly be solved for R_x . However, we can calculate the intersection point (if it exists) of the 45 degree line from the left side and the stepping function from the right side. The latter may be evaluated iteratively by

$$\begin{aligned} (R_x)_0 &= \sum_{i=1}^x C_i \\ (R_x)_k &= C_x + \sum_{i=1}^{x-1} C_i * \left\lceil \frac{(R_x)_{k-1}}{P_i} \right\rceil \quad (k = 1, 2, \dots) \end{aligned} \quad (3)$$

If after several steps $(R_x)_k = (R_x)_{k-1}$ and $(R_x)_k \leq D_x$ the test ends positively, the activity a_x is schedulable. The calculation has to be repeated for all activities.

In these days, (2) is the most frequently used test to predict the time behaviour of embedded systems. Over time it has received several extensions. If, for instance, higher priority activities are delayed for some reason (e.g. being blocked on a lock variable) these delay times have to be added to the computation time in order to model the worst case behaviour correctly. However, the more such extensions are included the more the test loses its precision. It may happen, that a test states deadline overrun, whilst in reality such an event never occurs.

Tabular Test

In many cases we want to model situations more realistically, with externally defined priorities, more complex than the simple one-server-station, or in any other way different from the systems hitherto assumed. In such situations we have to turn to more detailed scheduling plans [Fu 95]. A scheduling plan is a graphical diagram or tabular listing of the starting times, finishing times, preemption events and with respect to prediction eventually the time points, where deadlines are recognized to be missed. With periodic activities in order to catch all possible patterns of combination we have to analyse a time interval defined by the so called hyper period, that is the least common multiple of all periods present in the system. After that the patterns repeat.

The following example shows the essential features: We assume a system with two periodic activities.

Activity 1: period=10, ready time=1, busy time=3, deadline=8
 Activity 2: period=6, ready time=2, busy time=2, deadline=5

Activity 1 shall deliberately have higher priority. Activity 2 may be preempted. Note that we use the more general definition of a periodic activity with an initial delay (ready time). Fig. 2 indicates the events in the course of the hyper period.

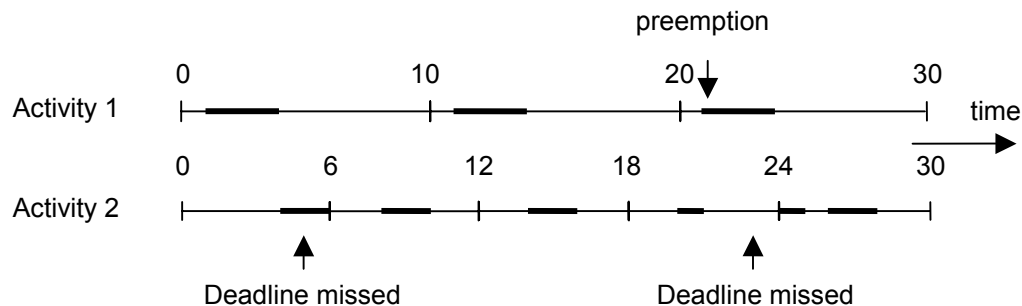


Fig. 2. Graphical illustration of a schedule

The scheduling plan will be laid down in a table similar to the one in Fig. 3. The example plan contains the time points of two deadline overruns (at time 5 and 23) and a preemption event (at time 21), these being examples of the greater power of the method.

Besides the time events the table contains additional useful information. For instance, it is possible to extract statistical measures of busy or idle phases. Even the so called priority inversions can be detected and analysed. Thus the tabular schedule can give much more help to the designer assuming the information is properly condensed and presented.

Time	Start Event	End Event	Preempt Event	Restart Event	Miss Event
1	1,1				
4	2,1	1,1			
5					2,1
6		2,1			
8	2,2				
10		2,2			
11	1,2				
14	2,3	1,2			
16		2,3			
20	2,4				
21	1,3		2,4		
23		1,3			2,4
24				2,4	
25		2,4			
26	2,5				
28		2,5			

Fig. 3. Tabular representation of a schedule (number pairs indicate activity and its repetition)

Additional Resources

- [LL73] C. L. Liu and J. W. Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, **20**, 46-61, (1973)
- [Fu 95] S. Fuchs, Prozessorzuteilung unter harten Realzeitanforderungen in dedizierten Systemen, PhD Thesis, University of Karlsruhe, (1995)

Vector Worldwide

France

Vector France SAS

168, Boulevard Camélinat
F-92240 Malakoff

Tel.: +33 1 4231 4000

Fax: +33 1 4231 4009

<http://www.vector-france.com>

Germany

Vector Informatik GmbH

Ingersheimer Str. 24
D-70499 Stuttgart

Tel.: +49 711 80670 0

Fax: +49 711 80670 111

<http://www.vector-informatik.com>

Japan

Vector Japan Co., Ltd.

Nishikawa Bldg. 2F
3-3-9 Nihonbashi, Chuo-ku
J-103-0027 Tokyo

Tel.: +81 3 3516 7850

Fax: +81 3 3516 7855

<http://www.vector-japan.co.jp>

Scandinavia

VecScan AB

Fabriksgatan 7
SE 412 50 Göteborg

Tel.: + 46 317 990 135

Fax: + 46 371 990 305

<http://www.vecscan.com>

USA

Vector CANtech, Inc.

Suite 550
39500 Orchard Hill Place
USA- Novi, Mi 48375

Tel.: +1 248 449 9290

Fax: +1 248 449 9704

<http://www.vector-cantech.com>