About Bounds of Buffers Shared by Periodic Tasks : the IRMA project

J. Legrand, F. Singhoff, L. Nana, L. Marcé
LIMI/EA 2215, University of Brest
20, av Le Gorgeu
29285 Brest Cedex
{jlegrand,singhoff,nana,marce}@univ-brest.fr

F. Dupont, H. Hafidi
TNI-Valiosys
Z.I. Pointe du Diable, BP 70801
29608 Brest Cedex
{Francois.Dupont, Hicham.Hafidi}@tni-valiosys.fr

Abstract

This paper presents the first results of the IRMA project. In this project, we study the feasibility of real time monitoring applications. By monitoring applications, we mean applications which detect and analyse failure information, and finally, present them to the user at maintenance periods. We suppose that these applications are composed of periodic tasks and of tasks which are randomly activated when faults occur. We also suppose that these two task families share resources and specially buffers. An application is feasable if all tasks meet their temporal constraints and if no buffer overflow exists.

In this paper, we propose tests to bound buffers shared by periodic tasks. These tests are implemented in a real time scheduling simulator which provide most of classical uniprocessor real time scheduling feasibility tests.

1 Introduction

Due to the increasing complexity of large real time systems, their maintenance becomes a crucial point.

This kind of system can be made of many different equipments. Therefore, the probability that one of the equipments fails can be high. Most of the time, large real time systems require regular maintenance periods.

In order to decrease maintenance cost, more and more often, large real time systems propose monitoring services (ie. aeronautical systems [2]). These services help operators to detect and diagnose failures. They provide features to collect data during the life of the system, to analyze them and to store analysis results into a persistent memory. Finally, they also provide tools for analysis information display at maintenance time.

We suppose that the studied monitoring applications are composed of two kinds of tasks. Some tasks are periodically activated and the others are activated on failure information. Tasks have to meet temporal constraints and are scheduled according to a real time scheduler such as Rate Monotonic[6].

Tasks are independent even if they share buffers. Buffer producers and consumers are not synchronized. Tasks produce or consume at their own rate : a periodic consumer can be awoken without having a message to read. Indeed, we suppose that only one message can be produced or consumed during a periodic task activation.

This paper relates the first stage of the IRMA project where we suppose that failure information is provided in a periodic way. All the tasks are therefore independent periodic tasks[6].

In IRMA, we study the feasibility of such applications. An application is said feasable if tasks meet their temporal constraints and if no buffer overflow occurs.

Task temporal constraints and buffer bounds are studied independently.

Task temporal constraints are checked with classical real time uniprocessor scheduling feasibility tests such as bound on processor utilization factor [6] and task response times [4, 1].

We propose buffer bounds which are built without assumption on the way the tasks are scheduled but with the assumption that every task meets its temporal constraints. In other words, bounds are valid even if the scheduling at execution time is different from the one planned at specification time (the differences may be due to unpredictable events such as jitter on task wake up time or task variable execution time, ...) provided that task deadlines are met.

In the second section of this paper, we show how to bound buffers shared by periodic tasks. These suggestions are illustrated with a tool presented in the third section. In the fourth section, we conclude and present ongoing works.

2 Bounds on buffers shared by periodic tasks

Let now suppose a set of tasks sharing buffers. Each of these tasks produces or consumes one message per periodic activation. Producer and consumer are not synchronized : they produce or consume at their own rate. A consumer can therefore be awoken even if no message is present in its buffer. In this case, we assume that the periodic activation of the consumer is discarded. Buffers

are managed in a FIFO manner.

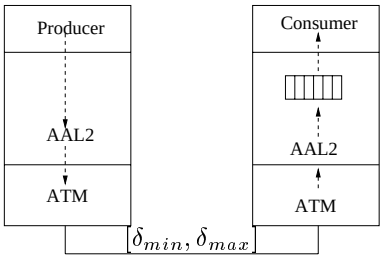

Figure 1: About the ATM AAL2 layer

To find a buffer bound, we can extend a result from the voice transmission service provided by the AAL2 layer of ATM networks.

In AAL2/ATM, a producer sends audio packets at a fixed rate d. This troughtput is expressed in cells per second, the protocol data unit of ATM networks. A bounded variable delay is needed to each cell to go from the sender to the receiver.

In an AAL2 communication service, the consumer should receive the cell flow in the same rate the producer sends it. To solve this problem, a buffer is added at the receiver side (see figure 1). Each received cell is then buffered during a sufficient amount of time to hide the variable transmission delay.

In [3], it is shown that a buffer bound B of :

$$B = \left\lceil \frac{\delta}{d} \right\rceil \tag{1}$$

allows to avoid buffer overflow. In (1), $\delta = 2.(\delta_{max} - \delta_{min})$ where δ_{max} and δ_{min} are respectively the maximum and the minimum transmission delay in the ATM network.

δ is the maximum time spent by a cell in the buffer. In the sequel, we call this delay **the maximum memorization delay. This delay is also the maximum delay between two successive consumptions.**

2.1 Applying AAL2 buffer bound to periodic tasks

The systems studied in the context of the IRMA project are similar to the one described above and we will show how to apply equation (1) to a set of periodic tasks sharing a buffer.

Later on, we will note P_i, the period of a task and D_i, its deadline. Obviously, if a buffer bound exists, the following equation holds :

$$\sum_{prod \in PROD} \frac{1}{P_{prod}} \leq \sum_{cons \in CONS} \frac{1}{P_{cons}} \tag{2}$$

where $PROD$ (resp. $CONS$) is the set of producer tasks (resp. consumer tasks) of a buffer and P_{prod} (resp. P_{cons}) the period of the producer *prod* (resp. of the consumer *cons*).

Equation (2) simply states that a bound exists only if the production rate is less than or equal to the consumer rate. We assume in this paper that one message is produced or consumed on each activation of a periodic producer or consumer.

Let see now how to apply the ATM/AAL2 bound when a buffer is shared by N producers and 1 consumer.

We first look for the maximum memorization delay of a message in the buffer. We study messages added to the buffer by a producer i. When a message of i is added to the buffer, y messages could be already present in the buffer. These y messages have to be consumed before the message of i since the buffer is managed in a FIFO way. Furthermore, in the worst case, i produces a message immediately after the beginning of a consumer activation. This message could be consumed at the end of a consumer activation. Then, the maximum memorization delay of a message of i is $\delta = P_{cons} + y.P_{cons} + D_{cons}$, or $\delta = (y+1).P_{cons} + D_{cons}$.

From the maximum memorization delay, we can now propose a buffer bound B :

$$B = \max_{\forall y \geq 0} \left(\sum_{prod \in PROD} \left\lceil \frac{\delta + O_{prod}}{P_{prod}} \right\rceil - y \right) \tag{3}$$

Equations (1) and (3) differ on four points. 1) Contrary to ATM/AAL2, N periodic tasks produce messages in IRMA, then, equation (3) have to count the messages produced by all tasks. 2) Contrary to ATM/AAL2, in (3), no particular assumption is done on task wake up times. For a producer *prod*, O_{prod} is the bound on the delay between the wake up time of the consumer and the wake up time of *prod*. Since during O_{prod}, *prod* can produce messages that we can not forget, O_{prod} have to be added to δ. 3) y messages are substracted from the bound expression (3). Indeed, contrary to ATM/AAL2 where no consumption is done during δ, in (3) y consumptions are done during this delay. 4) Finally, to find a buffer bound from (3), we have to study it for all positive integer values of y. However, we will see in the next section that simple bound expressions without y exist.

2.2 Buffer bound with $\forall i : D_i \leq P_i$

From the bound expression given above, we show how to find a simple bound when the deadlines of the tasks are equal to their periods. In this case, the maximum memorization delay is $\delta = (y+2).P_{cons}$ (since $D_{cons} = P_{cons}$).

With N producers and 1 consumer, in the worst case, the delay between the consumer and a producer wake up time is bounded by the period of the producer. Then,

$\forall prod \in PROD : O_{prod} = P_{prod}$. Since $\delta = (y+2).P_{cons}$, substituting from (3), we have :

$$B = \max_{\forall y \geq 0} \left(\sum_{prod \in PROD} \left\lceil \frac{(y+2).P_{cons} + P_{prod}}{P_{prod}} \right\rceil - y \right) \quad (4)$$

From equation (4), we can prove the following theorem :

Theorem 1 (a simple bound) *The bound of a buffer shared by 1 periodic consumer and N periodic producers, when $\forall i : D_i \leq P_i$ and when all deadlines are met, is :*

$$B = 2.N \quad (5)$$

if tasks are harmonics and

$$B = 2.N + 1 \quad (6)$$

in the other cases.

Proof :

Let prove the buffer bound for non harmonic tasks. From (2) and (4), we will show that, in the worst case, one producer can add 3 messages in the buffer and the $N-1$ other producers can only add 2 messages.

Let show that a producer add $y+3$ messages during the memorization delay.

From (2), if a task set is composed of 1 consumer and N producers, $P_{prod} > P_{cons}$ holds. The maximum throughput of a producer i can be achieved when $P_i \to P_{cons}$. In this case $\left\lceil \frac{(y+2).P_{cons}+O_i}{P_i} \right\rceil = \left\lceil \frac{(y+2).P_{cons}+P_{cons}}{P_{cons}} \right\rceil = y+3$ messages can be produced during δ.

Now, let find I, the maximum time interval of P_i within which i can produce up to $y+3$ messages.

If $P_i \to P_{cons}$, the first (or the last) of the $y+3$ messages produced by i is done during the time interval $]0, P_{cons}[$. This delay can be uniformly distributed to the $y+1$ producer activations ; in order to maximize its period with $y+3$ produced messages.

The longest period of i with $y+3$ produced messages is then $P_i = \frac{P_{cons}}{y+1} + P_{cons} = \frac{(y+2).P_{cons}}{y+1}$. We have shown that a producer can add $y+3$ messages in the buffer when its period belongs to $I =]P_{cons}, \frac{(y+2).P_{cons}}{y+1}[$. Let us consider the $N-1$ other producers now.

If the period of the producer i is close by $\frac{(y+2).P_{cons}}{y+1}$, then, the $N-1$ other producers have period near to $(y+2).P^+_{cons}$. Indeed, (2) can be rewritten as follows :

$$\frac{1}{x} + \sum_{prod \in PROD^*} \frac{1}{P_{prod}} \leq \frac{1}{P_{cons}}$$

or :

$$\sum_{prod \in PROD^*} \frac{1}{P_{prod}} \leq \frac{x - P_{cons}}{x.P_{cons}}$$

where $PROD^*$ is $PROD - \{x\}$ (x is the producer which adds 3 messages).

Let assume $f(x)$:

$$f(x) = \frac{x - P_{cons}}{x.P_{cons}}$$

and investigate it when $x \to \frac{(y+2).P_{cons}}{y+1}$ and $x \to P_{cons}$:

- $\lim_{x \to \frac{(y+2).P_{cons}}{y+1}} f(x) = \frac{1}{(y+2).P_{cons}}$
  Or $\sum_{prod \in PROD^*} \frac{1}{P_{prod}} \leq \frac{1}{(y+2).P_{cons}}$,
  which implies, in the worst case :
  $\forall prod \in PROD^* : P_{prod} \to (y+2).P^+_{cons}$.
- $\lim_{x \to P_{cons}} f(x) = 0$
  Or $\sum_{prod \in PROD^*} \frac{1}{P_{prod}} \leq 0$, and thus, $\forall prod \in PROD^* : P_{prod} \to \infty$.

From the investigation of $f(x)$, we have shown that producers from $PROD^*$ have a period included in $J =](y+2).P_{cons}, \infty[$. Then, the largest production message is

$$\sum_{prod \in PROD^*} \left\lceil \frac{\delta + O_{prod}}{P_{prod}} \right\rceil$$

or :

$$\sum_{prod \in PROD^*} \left\lceil \frac{(y+2)P_{cons} + (y+2).P^+_{cons}}{(y+2).P^+_{cons}} \right\rceil = 2.N - 2$$

From (4), the buffer bound is :

$$B = y+3+ \sum_{prod \in PROD^*} \left\lceil \frac{(y+2)P_{cons} + (y+2).P^+_{cons}}{(y+2).P^+_{cons}} \right\rceil - y$$

and finally : $B = y + 3 + 2.N - 2 - y = 2.N + 1$

Let see now the case where tasks are harmonics. The method previously applied for non harmonic tasks could be used in this case except that with harmonic tasks, $y+2$ messages are produced during the memorization delay. Indeed, when two tasks i and j have the same period, $O_i = O_j = 0$. In the worst case, a producer i sends $\left\lceil \frac{(y+2).P_{cons}}{P_{cons}} \right\rceil = y+2$ messages. During a time interval I, P_i can take values in $[P_{cons}, \frac{(y+2).P_{cons}}{y+1}]$. Substituting the information above in (3) yields :

$$B = y+2+\sum_{prod \in PROD^*} \left\lceil \frac{(y+2)P_{cons} + (y+2).P_{cons}}{(y+2).P_{cons}} \right\rceil - y$$

and finally : $B = y + 2 + 2.N - 2 - y = 2.N$

3 The Cheddar real time scheduling simulator

From the ATM/AAL2 buffering service, we have proposed bounds on buffers shared by periodic tasks. We now describe a tool we use to test these bounds and which can help to model an application and to analyse its feasibility.

This tool, Cheddar, provides services to check temporal constraints of tasks. Cheddar implements bounds similar to the ones presented in the previous section and some classical uniprocessor scheduling feasibility tests.

Cheddar is a tool published under the GNU/GPL license. Source code, binaries and documentation can be downloaded from http://beru.univ-brest.fr/~singhoff/cheddar. Cheddar runs on Solaris, Linux and Win32 systems.

The tool is composed of two independent parts :

- A graphical editor used to design a real time system or a real time application.
- An object oriented framework providing the most usual real time schedulers and feasibility tests.

Cheddar lets you describe a system or an application composed of periodic/aperiodic tasks, processors, buffers, shared resources and messages.

From an application description, Cheddar provides two features : feasibility tests and a simulation engine.

The main feasibility tests provided are response time and processor utilization factor computation for Rate Monotonic, Deadline Monotonic, Earliest Deadline First and Least Laxity First [6, 4]. Of course, Cheddar can bound buffer size with equations such as (5) and (6).

Simulation features allow the user to display a scheduling in a graphical way. From this simulation, much information can be computed by Cheddar : response times, shared resource blocking times[1], free time units ... Today, most of the classical scheduling algorithms are implemented in Cheddar but the simulation engine can be tuned by users with a small and easy to use language. This is a useful feature since in many cases, no feasibility tests are available for specific task models or scheduling algorithms.

An exhaustive list of services provided by Cheddar is given on the Cheddar user's guide Web pages.

[1]Supported shared resource protocol are PCP and PIP.

4 Conclusions and ongoing works

This paper have presented the IRMA project. In this project, we have studied feasibility of real time monitoring applications. Applications we have considered run on a uniprocessor and are composed of tasks which can be activated in a periodic or in a random manner. This two task families share resources such as buffers. Checking feasibility of such applications consists in checking that tasks meet their deadline and that no buffer overflow occurs.

In this paper, we first supposed that failure information was provided periodically to the monitoring application. All tasks are then periodic ones.

We have proposed a test to find bounds on buffers shared by periodic tasks. These bounds do not suppose that tasks are scheduled according to a particular scheduler : they are valid for all scheduling provided tasks meet their deadline.

The proposed bounds are implemented in a GPL real time scheduling simulator : the Cheddar simulator. This simulator provides most of classical uniprocessor feasibility tests for real time schedulers (fixed priority and deadline driven schedulers).

Future works in the IRMA project should now consider monitoring applications where failure information is provided in a random way. Indeed, most of the time, reliability of embedded equipments is expressed by random law[5]. Unfortunately, few feasibility tests are proposed when periodic and randomly activated tasks share a processor and buffers.

References

[1] A. N. Audsley, A. Burns, M. Richardson, and K. Tindell. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, pages 284–292, 1993.

[2] B. Burchell. A3XX Maintenance : A First Look. *Overhaul and Maintenance Revue. URL : www.aviationnow.com*, August 2000.

[3] M. Gagnaire and D. Kofman. *Réseaux Haut Débit : réseaux ATM, réseaux locaux, réseaux tout-optiques.* Masson-Inter Editions, Collection IIA, 1996.

[4] M. Joseph and P. Pandya. Finding Response Time in a Real-Time System. *Computer Journal*, 29(5):390–395, 1986.

[5] C. M. Krishna and K. G. Shin. *Real-Time Systems.* Mc Graw-Hill International Editions, 1997.

[6] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.