An Evaluation and Simulation Technique of Real-Time Operating Systems

Frank Golatowski, Dirk Timmermann, Kai Pankow
{gol,dtim**}**@baltic.e-technik.uni-rostock.de
University of Rostock
Department of Electrical Engineering
Institute of Applied Microelectronics and Computer Science
Richard-Wagner-Str. 31
18119 Rostock- Warnemünde
Germany

Abstract
This paper aims at the performance evaluation of real-time computer systems and the simulation of typical real-time applications using a synthetic benchmark program. Our approach is based on the Hartstone Uniprocessor Benchmark. We have implemented all test series using C for real-time UNIX operating systems. Using the benchmark it is possible to observe the impact of increasing load on the *overall* system in a real-time sense. We give results obtained running the Lynx real-time Operating system on a Pentium PC.

1 INTRODUCTION

There are numerous test suites for real-time operating systems. Three general methods have been devised, differing in the degree of detail at which they evaluate the target system:
a) fine-grained benchmarks
b) application-oriented benchmarks
c) simulation-based evaluations
Using a fine-grained benchmark one can evaluate system specific performance numbers, like context switches, interrupt and task response. There is no standard method available for determining performance numbers. Only in early drafts of the real-time POSIX Standard 1003.1b some proposals for determination of system times are given [4].
The best well-known fine-grained benchmark is the Rhealstone Benchmark [9]. This benchmark defines six performance numbers and programs are given to measure these values. The sum of these values is used as a performance index for a real-time operating system. As the various values do not evenly affect the Rhealstone number it is not a good representation of the measured real-time operating system. Therefore the calculation of a weighted Rhealstone number has been proposed. Although a total Rhealstone number can not represent a complex real-time operating system at all situations the different values determined by the Rhealstone Benchmarks give a general idea about the average behavior of the system kernel.
Application-oriented benchmarks analyze the underlying real-time system employing specific applications. The best-known one in this area is the Hartstone Uniprocessor Benchmark [12][3]. Another one covers a set of specific parameterized benchmarks and provides typical workloads for embedded control application [10].
The Hartstone Benchmark defines some sets of requirements typical for real-time applications. In the experiments derived from real-time scheduling theory a real-time system is stressed to find out whether the given system meets hard real-time requirements. Hartstone as understood by the inventors „is a system requirement rather than an implemented program“. Currently, only a small subset of test series has been implemented in ADA [3].
Simulation-based evaluations model a real-time system under various restrictions and up to a specific level of detail, and then implement and run the model. The disadvantage of this method is that it is impossible to account for all effects in an actual real-time system. The advantage of this approach is that the hardware itself can be modeled in conjunction with software.
Also combinations of the three methods are possible.

2 SCHEDULING THEORY

In parallel to measurement technique software analysis of real-time systems becomes more and more important in the field of performance evaluation. In the meanwhile advanced engineering methods are available which can give information whether a real-world system meets its timing requirements. Here we state some important research results in real-time analysis and scheduling theory.
There has been a lot of work in this field. Two research directions can be identified. The first one concentrated on determining the feasibility a given process set with rate-monotonic analysis and the appropriate model introduced by Liu and Layland [11], the second one is based on deriving worst-case response times.
In [11] some assumptions are made:
- all processes are periodic
- all processes have deadlines equal to their period
- all processes are independent
- all processes have a fixed computation time

For those processes whose priorities are assigned to tasks in rate-monotonic order a process set is schedulable if the following condition holds:

$$n(2^{\frac{1}{n}} - 1) \geq \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad (1)$$

(n...number of processes T_j...period of the i-th process C_i ...computation time of i-th process)

For large values of n the upper bound of utilization is 69,31%; for n=5 processes the limit is 74,34 %. That is, if the utilization of five processes is less than 74,34 % all deadlines are met. Note this is a sufficient condition and the bound is conservative, that means a process set with a utilization greater than 74,34 % could be possibly scheduled on one processor.

Response times analysis initiated by Harter [6] can be readily used for determining exact feasibility of a process by comparing its worst-case response time to its deadline.

$$R = C_i + \sum_{i \in H} \left\lceil \frac{R}{T_j} \right\rceil C_j \qquad (2)$$

(n...number of processes T...period of the i-th process C_j*...computation time of i-th process)*

R is the response time of process τ_j and *H* is a set of processes with priorities higher than the priority of process *i*. The process τ_j is feasible if for any value of response time $R \in [0, D_i]$ (D_j*...deadline of the i-th process)* the above condition holds. The smallest value equals the worst-case response time of process τ_i. In this form the equation is hard to solve. So this equation can be solved by a recurrence relation [1][7].

$$R_i^{n+1} = C_i + \sum_{i \in H} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j \qquad (3)$$

where R_i^n is the response time in the *n*th iteration and the required response time is the smallest value of R_i^{n+1}. To calculate response times, it is necessary to compute R_i^{n+1} iteratively until the first value *m* is found that satisfies $R_i^{m+1} = R_i^m$. The resulting response time now is R_i^{m1}. If this value is equal to or smaller than its deadline then process τ_i meets its deadline.

For a more detailed description of real-time scheduling theory see [8].

According to Audsley et al. [2] there are three important results of real-time scheduling theory and may be recognised as a basis for the majority of research in this field:

1. sufficient and necessary feasibility tests
2. analysis of process interaction
3. inclusion of aperiodic process

These points are addressed by the execution of the following benchmark.

3 PERFORMANCE EVALUATION OF REAL-TIME SYSTEMS

The measurement of fine-grained values is important but in a real-time environment influences from the whole system should be considered. One attempt to evaluate the *overall* performance of a real-time system is the Hartstone Uniprocessor Benchmark. Our method is based on the definitions given by the Hartstone Benchmark. We have implemented all testseries of the benchmark using C for real-time UNIX systems. The plattform is the real-time UNIX operating system LynxOS Version 2.3 running on a Pentium PC (90MHz). This implementation aims at the observation of the system behavior when increasing load. Due to its flexibility it also permits the execution of optional synthetic applications.

The model of the benchmark focuses on typical applications for real-time systems. The benchmark has five different test series consisting of several experiments. Each series consists of different processes: periodic, aperiodic and synchronization processes. Periodics have hard deadlines determined by their period; aperiodics may have hard or soft deadlines. Aperiodic processes with soft deadlines run as background processes. Missing a hard deadline in a real system means that the consequences may be catastrophic. The system may tolerate the missing of soft deadlines.

Experiments start with a baseline process set characterized by a number of processes, their priority, their synthetic load, process period, starting time and interarrival time. The workload is increased and the system is observed, whether the deadlines are met. The results given from the benchmark may be used for a closer view of real-time scheduling theory. On a real-time system timing constraints have to be held. In the model used the timing constraint for periodic processes is the period; each process must be executed within its period. If the workload is too high and a process is interrupted by higher prioritized processes a process can miss this timing constraint; it misses its deadline. On a real system a system overhead coexists for example due to context switches, scheduling overhead, and kernel delay.

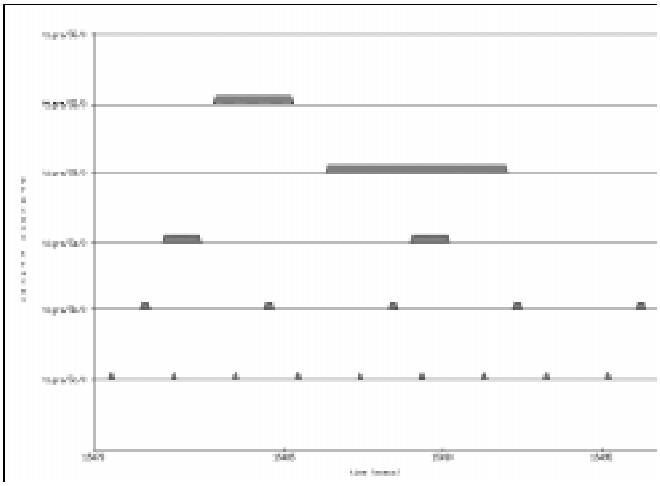

Figure 1: Execution of periodic harmonic processes

The Hartstone Uniprocessor Benchmark defines some experiments with different series. It considers not only harmonic and non-harmonic periodic processes but also aperiodic, sporadic, and synchronization processes.

Each experiment starts with a baseline process set. A baseline process set consists of at least five processes. Each process executes a definite workload. The synthetic workload consists of small portions of the well-known Whetstone-Benchmark. Each process executes this synthetic load in a loop according to a parameter within a test description file. The benefit of this load is that granularity of the Small-Whetstone load is much finer than the Whetstone instructions in their whole.

The workload derived from the Whetstone benchmark is measured in Kilo-Whetstone Instructions (KWI), because this synthetic load consists of thousand instructions of the Whetstone benchmark. The load executed during its activation is measured in Kilo-Whetstone instructions per period (KWIPP) and the executed load during one second is measured in Kilo-Whetstone instructions per second (KWIPS). The different series of an experiment are resulting from increasing workload. Increase of workload is achieved by increasing the frequency of one process, of all processes, the amount of the executed synthetic workload and additional processes that have to be executed. Table 1 shows the full set of the Hartstone experiments. The derivation of the Hartstone experiments is based on real-time scheduling theory, see [12].

Table 1: Hartstone Uniprocessor Benchmarks experiments [12]

Experiments / series	expr1	expr2	expr3	expr4	expr5	expr6
PH	$T_5 \downarrow$	$T_1 - T_5 \downarrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		
PN	$T_5 \downarrow$	$T_1 - T_5 \downarrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		
AH	$IAT_{sp} \downarrow$	$C_{sp} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$	$IAT_{ap} \downarrow$	$C_{ap} \uparrow$
SH	$C_{s\text{-}within} \uparrow$	$C_{s\text{-}outside} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$	$n_s \uparrow$	
SA	$IAT_{sp} \downarrow$	$C_{s\text{-}within} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		

T_i ...period of i-th process
C_i ...computation time of i-th process
C_{sp} ...computation time of the sporadic process
C_{ap} ...computation time of the aperiodic process
$C_{s\text{-}within}$ ...computation time of the server within its critical section
$C_{s\text{-}outside}$ ...computation time of the server outside its critical section
IAT_{sp} ...interarrival time of sporadic process
n_{Task3} ...number of tasks with characteristics of the third task
n_s ...number of server tasks

In our experiments different kinds of workload are increased step by step. If deadlines are missed for the first time at a given rate the *breakdown utilization point* is reached. With our implementation we can observe the system beyond the breakdown utilization, i.e. we inspect the system behavior under overload conditions.

The first series of experiments (PH-series; PH stands for periodic and harmonic) starts with a baseline process set (the process sets are described in Table 2 consisting of five independent periodic processes with harmonic frequencies. Figure 1 shows these five periodic and harmonic processes. This is a screenshot made with the Timescan utility. (Timescan comes with PosixWorks of LynxOS Vers. 2.3. We use this tool for the visualization of the running process sets.) Harmonic process means that the frequencies of the processes are a multiple of the smallest frequency. According to Table 1 the load of the system is increased in the different experiments and the real-time system is observed. The workload is increased by varying only one parameter of a series, and holding the other ones constant.

In the PH-series the workload is increased by

- decreasing the period of the process with the highest priority (process 5 of this series) (PH-1)
- decreasing the periods of all processes (PH-2)
- increase the workload of all periodic processes (PH-3)
- add a new process with the same characteristic as the third periodic (PH-4)

In the PN-series (periodic non-harmonic) characterized by non-harmonic periodic processes the workload is increased in the same manner.

As noted above in our experiments the influence of aperiodic processes is examined. In the experiments of the AH-series sporadic processes and background processes are observed. The aperiodics are characterized by their random activations. They are subdivided into sporadic processes with hard deadlines and background processes which have soft deadlines. The deadlines are determined by their arrivals. In order to meet hard deadlines of sporadic processes it is necessary to introduce a minimal interarrival time. So the system may handle bounded arrivals with hard deadlines. In these experiments the observation is focused both on meeting deadlines of hard real-time processes and on response time of the background process.

The influence of interprocess communication is analyzed in the SH-series. In these experiments we use a process set with smaller workload (see Table 4) than in the PH-experiments because synchronization leads to poor utilization at the breakdown utilization point. Based on this baseline process set at least one server process is added with which each periodic process has to synchronize. We have realized this synchronization by semaphores. To observe the behavior with synchronization other mechanisms can be added.

As an another example of a series we consider the SA-series of the benchmark here. This is the most complex series. Each experiment starts with a baseline process set. This series consists of periodic, sporadic, aperiodic, and a server process. All the periodic processes have to synchronize once with the server process during their period. The server executes the workload within and outside its critical section. The SA-series includes all types of processes of the other series.

In the several experiments of the SA-series the workload is increased by

- decreasing the interarrival time IAT_{sp} of the sporadic process (SA-1)
- raising up the workload of the server within the critical section (SA-2)
- raising up the workload of all periodic processes (SA-3)

- adding a new process with the same (SA-4) characteristic as the third periodic

4 EXECUTION OF EXPERIMENTS

Each experiment of a series starts with a baseline process set. This baseline process set is equal to all series but in the various tests of an experiment one parameter of a series is changed while keeping the other constant. The process set is specified in a test description file.
At the beginning of each experiment the raw performance executed within one process is measured. This is the maximum value of the workload achievable if the system executes one process. The raw performance is measured in KWIPS. Thereafter the test description file is interpreted. It starts with the specification of the baseline process set. According to the specifications processes are created and started. An experiment terminates after a definite amount of time or a definite amount of missed deadlines.
The approach implemented in the benchmark is based on a real-time clock and an appropriate RTC-driver software. Through this driver necessary timing functions are available. The driver manages the enqueuing of process requests and passing them to the scheduler.
The main analysis method is described as follows. Given that a deadline can not be met the next will be omitted. The omitted deadline counts as not met. Using this policy the system can serve at least the following requests.

Note that on various machines a process set may differ because the raw performance characteristics are different. This process set is characterized by high frequencies. It produces approximately 30% of raw performance workload. In the corresponding experiments this basic load will increase to detect the breakdown utilization point and the behavior of the system under overload conditions. Processes get a priority according to the rate-monotonic order: the higher the frequency of a process (rate) the higher the priority. For example the first periodic process P1 exhibits the lowest priority. Its workload is 64 Kilo-Whetstones per period. Process P5 is assigned the highest frequency of all periodic processes. The processed workload per time unit is equal for all periodic processes.
In Table 3 it is shown that the sum of the workload of both aperiodic processes (background process B1 and sporadic process S1) is approximately equal to the periodic processes. Note the frequencies given are used for generating random values so that the activation of the aperiodic processes is irregular.
The characteristic of the server (Y) results from the necessary number of activations of the periodic processes, because all periodic processes have to synchronize with the server. In the SA-experiments we use a process set with smaller frequencies because the behavior of this series is affected by synchronization effects.

Table 2: Definition of baseline process set for PH-series on the reference Pentium system

	start time (sec.)	duration (sec.)	priority LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
P1	5	10	21	32.00	64	2048
P2	5	10	22	64.00	32	2048
P3	5	10	23	128.00	16	2048
P4	5	10	24	256.00	8	2048
P5	5	10	25	512.00	4	2048
Σ	5	10		992.00		10240

Table 3: Aperiodic processes of the baseline process set

	start time (sec.)	duration (sec.)	priority LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
B1	5	10	20	32.00	32	1024
S1	5	10	26	512.00	2	1024
Σ						2048

Table 4: Baseline process set of the SA-series

	start time (sec.)	duration (sec.)	priority LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
P1	5	10	21	1.00	3071	3071
P2	5	10	22	2.00	1535	3070
P3	5	10	23	4.00	767	3068
P4	5	10	24	8.00	383	3064
P5	5	10	25	16.00	191	3056
B1	5	10	20	1.00	1536	1536
S1	5	10	26	16.00	192	1536
Y	5	10	27	31.00	1	31

The benchmark has been executed on a 90MHz Pentium Processor, with 512 KB secondary cache and 32 MB of main memory. The used compiler is GNU gcc V.2.6.
The raw performance measured at the beginning of each experiment is 33470 KWIPS.
The diagrams [Figure 2-5] show the measured utilization (right y-axis), the missed deadlines (left y-axis), and the nominal workload on x-axis. The measured utilization is based upon the raw performance measured when only one process executes the synthetic load. The nominal workload is the ratio between raw performance and theoretical workload the system has to execute. The theoretical workload results from the test description. The measured utilization is the ratio between measured workload and raw performance. All values are given in percent.

The utilization is equal to the nominal workload up to the breakdown utilization point. Beyond that point the measured utilization is smaller than the nominal workload because missed deadlines result in omitted requests. (See the utilization curve in the diagrams.)

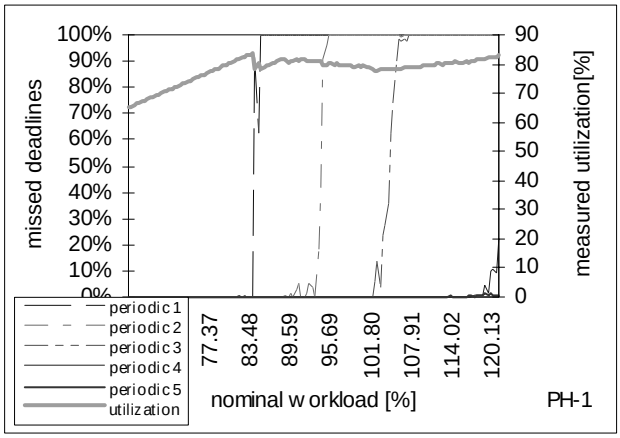

Figure 2: Utilization of PH-1 tests

The measured breakdown utilization point of the PH-1 experiment shown in Figure 2 is 81%. Note in this experiment only the frequency of the fifth process are increased.and at the BU point its frequency reaches 1152 Hz. In PH-3 experiments the breakdown utilization point has the highest value (88%) of all series. In an ideal system without context switching time the theoretical maximum utilization would be 100%. So in this experiment the influence of system overhead can be seen, as the number of activations remains constant. Only the computation time is increased. So here the maximum utilization of all experiments is possible.

The PH-experiments demonstrate that processes miss their deadlines according to priority; at first the lowest prioritized process misses its deadlines. For all processes a sudden missing of all deadlines can be seen. If one process misses its deadlines the measured utilization decreases.

In Figure 4 and Figure 6 the influence of aperiodic processes is shown. The BU-point in experiment AH-2 is 65% because the highest priorized periodic process (P5) misses its first deadline at this point. It can be seen that at first high prioritized processes miss their deadlines because they are blocked by the sporadic one whose load is increased in this experiment block them. The sporadic process misses its deadlines at 80% because its workload is too big to hold a minimal separation time necessary for this kind of process. Beyond the breakdown utilization point any processes miss their deadline.

In the AH-5 experiment all deadlines are met. This is due to the fact, that only the load of the background process is increased which has the lowest priority and for this kind of processes the response time is analyzed. Whenever the benchmark detects that the aperiodic load is too high this load will not be applied. Starting at an utilization rate of 36% (basic load with all processes of the baseline process set) 90 % of the raw performance may be reached. The asymptotic curve is due to the exponential distribution of the aperiodic process.

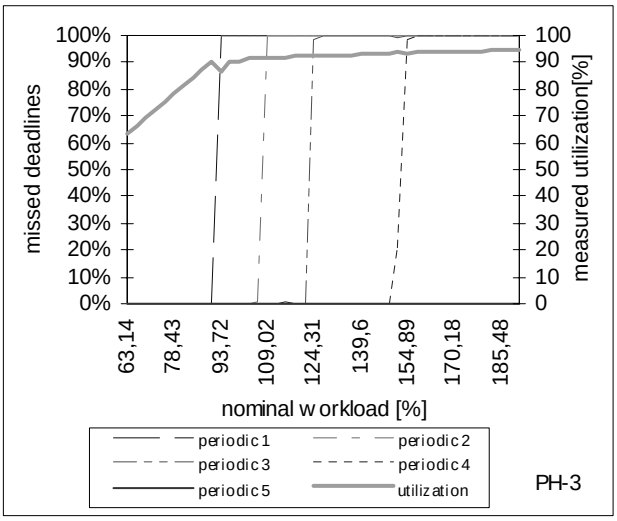

Figure 3: Utilization of PH-3 tests

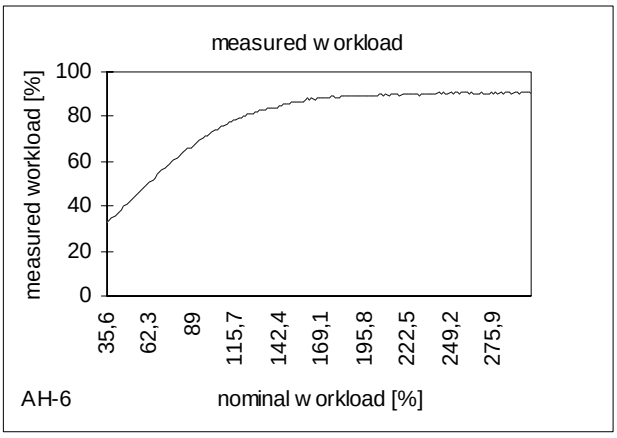

Figure 4: Total utilization of AH-6 tests

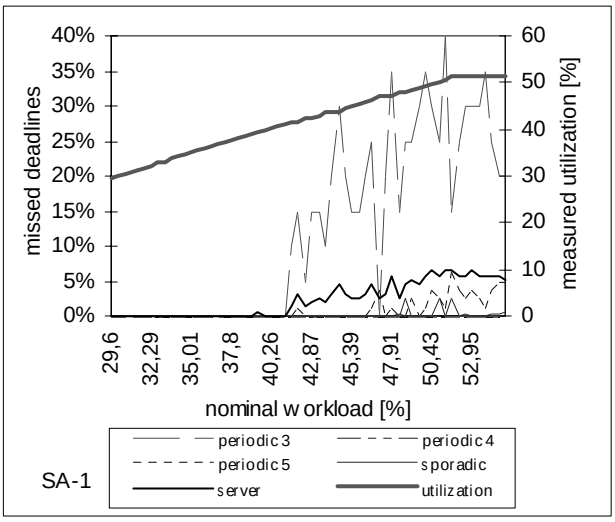

Figure 5: Utilization of SA-1 tests (Lynx-OS)

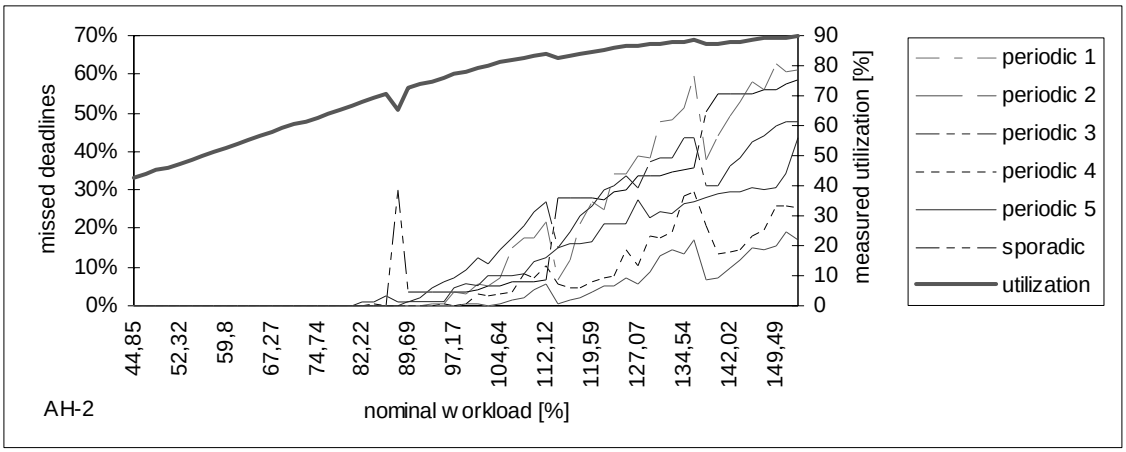

Figure 6: AH-2 experiment

In the SA-1 experiment the periodic processes have to synchronize with a server process. The server process has the highest priority and its activation count corresponds to that of all periodics. First the server misses its deadlines at an utilization of 40%. If a periodic process misses a deadline the server misses the deadlines, too, because server and periodic process are synchronized. (Figure 5)

All the given results are relative values based on the measured raw performance. This value must be taken into consideration as raw performance heavily depends on the compiler and compilation options.
An interesting fact is that the evaluated real-time operating system exhibits the same behavior if lower frequencies in the baseline process set are used.

5 CONCLUSION

In this paper, we show selected results of the evaluation of LynxOS using our implementation of the Hartstone Uniprocessor Benchmark. The benchmark enables us to find the breakdown utilization point of a system and to observe the behavior of a real-time system under overload conditions.
Applying the described method it is possible to find process sets representing the critical workload where the given system is still able to operate correctly in a hard real-time sense. Beyond that point any increased workload results in missed deadlines.
The program allows the description of real world applications in a process set and its execution. Due to this fact our implementation is suitable for the simulation of a wide range of application classes. The objective is to determine whether a real-time system will be able to meet the given requirements.

[1] Audsley,N., Burns,A., Wellings, A.J, "Deadline Monotonic Scheduling Theory and Application," Control Eng. Practice, vol. 1, no. 1, Feb.1993, pp. 237-250
[2] Audsley,N., Burns,A., Davis,R., Tindell,K., Wellings, A.J, "Fixed Priority Pre-emptive Scheduling: An Historical Perspective," Real-time Systems Journal, vol. 8, pp. 173-198, Kluwer Academic Publishers, 1995
[3] Donohoe,P., Shapiro, R., Weidermann, N., "Hartstone Benchmark user´s guide," Software Engineering Institute, Carnegie Mellon University, Technical Report CMU-SEI-90-TR-1, May 1990
[4] Gallmeister, B.O., "Programming for the Real World: POSIX.4" , O´Reilly & Associates, Inc., 1995
[5] Golatowski, F., Bösel, H., Paukert, A., "Implementierung eines applikationsorientierten Benchmarks für Echtzeit-Unix-Betriebssysteme, " in Echtzeit 95 Conference Proceedings, Rzehak, H., Ed. Karlsruhe, 1995, pp. 63-70
[6] Harter, P.K.: "Response Times in level-structured Systems, " Techn. Report, Univ. of Colorado, Boulder, 1984
[7] Joseph, M, Pandya,P., "Finding Response Times in a Real-Time System," Computer Journal J., vol. 29, no. 5, May 1986, pp. 390-395
[8] Joseph, M. (ed.), "Real-time Systems-Specification, Verification and Analysis, " Prentice Hall, 1996
[9] Fuhrt, B. et al: "Real- Time UNIX Systems," Kluwer Academic Publishers, 1991
[10] Lindmeier, H., Rauh, D., Viguier M.: „Benchmarking for Embedded Control and Real-Time Applications," Report, ESPRIT III - OMI, Project No. 6271, Dec. 1992,
[11] Liu, C.L. Layland, J.W, "Scheduling Algorithms for Hard Real-Time Environments," Journal of the ACM, vol. 20, no. 1, pp. 46-61, 1973
[12] Weidermann, N.H., Kamenoff, N.I., "Hartstone Uniprocessor Benchmark: Definitions and experiments for real-time systems, " Real-Time Systems Journal, vol. 4, no. 4, pp. 353-383, Kluwer Academic Publishers, 1992