DISTRIBUTION OF TIME INTERVAL BETWEEN SUCCESSIVE INTERRUPT REQUESTS

Wojciech Noworyta
Institute of Engineering Cybernetics - Wroclaw University of Technology

Abstract: In majority of computer systems, the most time-critical tasks are performed by interrupt service routines. In some cases pooling methods are utilized, especially when I/O hardware is not capable of stand-alone operation. When these two approaches are mixed together conflicts are likely. Pooling procedure may not tolerate delays caused by interrupts and interrupt-driven procedures may not be able to wait until pooling driver finishes its job. In various systems some of the time-critical operations can be repeated or skipped, if CPU fails to service them in time. Statistical methods can be used to verify if system performance satisfies requirements. To apply them the distribution function of time interval between two consecutive interrupt requests is calculated on the basis of a simple theoretical model. The model is then verified by empirical measurements.

Key words: interrupt, distribution function, streams merging, performance modeling.

1. INTRODUCTION

Observations of program execution performance have been taken using the most widespread hardware and operating systems (Windows 98/me/XP, Linux) and embedded processor MB91F362. In all cases a significant influence of operating system jobs and interrupt-driven activity has been detected. Asynchronous tasks particularly affect “smoothness” of execution making application response time random and unpredictable. These effects can be easily observed in low-end systems during multimedia playback. Methods of compensating IRQ introduced slowdowns may be applied to re-

store acceptable quality of real-time performance [1]. Since interrupts have the highest priority a vast majority of internal kernel functions must be able to tolerate unexpected delays during execution [2].

Execution of application while interrupts are enabled can be compared to queuing model where streams of requests are merged into one queue. When the queue is empty an application code is executed [3]. Queuing models with mixed type streams are not popular. Coexistence of periodical and random events results in very complex equations or problematic assumptions [4].

Many publications focus on scheduling algorithms that should give best overall quality of service of a real-time system [5]. The statistic nature of interrupts and other factors causing execution time uncertainty is rarely taken into account. Sometimes the simplest algorithms like Earliest Deadline First are suggested to be the most efficient solution [6].

In contrast to hard real–time systems where well-timed response must be assured soft real–time approach relays on statistical assurance of required performance [7]. Since average number of missed deadlines is not a good metrics n-out-of-m is proposed [8].

2. MODEL OF INTERRUPT REQUESTS

To investigate a nature of interrupt request process a spectral density of the empirically measured time between interrupts was plotted. Two components were observed: single periodical and the white noise. Such spectrum proves that interrupts can be categorized as:

- system timer interrupt
- random interrupts from other sources

It is known, that interrupts from peripheral devices are not 100% random. The observation of spectral plot shows that collection of all interrupts makes good approximation of the white noise in the majority of the cases.

Let's assume computer system where two types of interrupts are present. There is only one periodical timer interrupt and unlimited number of random interrupts. The nature of interrupts is as follows:

- Timer interrupt with frequency $f_x = 1/t_{x1}$ is requested periodically in fully deterministic manner.
- Peripheral interrupts are requested randomly according to Poisson process with parameter λ. If many sources of random interrupts are present, with intensities $\lambda_1, \lambda_2, \ldots \lambda_n$ that they can be described as one process with intensity $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_n$.

All interrupts are treated identically. The cumulative distribution function of time between two consecutive interrupt requests is to be calculated.

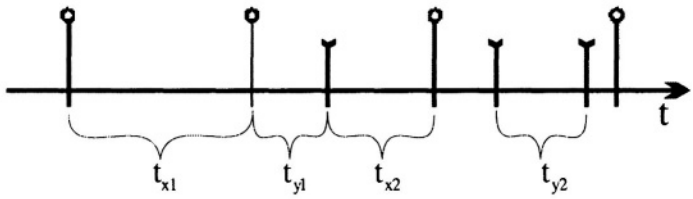

Figure 1. Interrupt requests and time intervals between them

Example interrupt requests and time intervals are presented on figure 1. Timer interrupts are marked with circle while random ones are marked with arrow-tail. The time interval can be measured between:

- two timer interrupts t_{x1}
- timer interrupt and random interrupt t_{y1}
- random interrupt and timer interrupt t_{x2}
- two random interrupts t_{y2}

Non-zero correlation between t_{y1} and t_{x2} is not taken into account since only distribution of time interval between (any) two interrupts is calculated.

2.1 Weighted sum of alternative distributions

To calculate the CDF of time between two consecutive requests two alternative cases should be considered:

- time interval begins after timer interrupt (t_{x1} or t_{y1}) with probability p_1
- time interval begins after random interrupt (t_{x2} or t_{y2}) with probability p_2

Cumulative distribution function is then calculated as a weighted sum of distribution functions of two alternative cases.

In sufficiently long time period $T \to \infty$ it is expected to appear $T*f_x$ timer interrupts and $T*\lambda$ random interrupts so the probabilities are:

$$p_1 = \frac{f_x}{f_x + \lambda} \quad \text{and} \quad p_2 = \frac{\lambda}{f_x + \lambda} \tag{1}$$

If the last interrupt was timer interrupt:

CDF of the interval t_{x1} from a timer interrupt to the next timer interrupt:

$$F_{x1}(t) = 1(t - t_{x1}) \tag{2}$$

CDF of the time interval t_{y1} from a timer interrupt to the nearest random:

$$F_{y1}(t) = 1(t)(1 - e^{-\lambda t}) \tag{3}$$

Distribution of the time interval from a timer interrupt to the next timer or random interrupt (whichever comes first):

$$F_1(t) = 1 - (1 - F_{x1}(t))(1 - F_{y1}(t)) \tag{4}$$

$$F_1(t) = \begin{cases} 0 & \text{for } t \le 0 \\ 1 - e^{-\lambda t} & \text{for } 0 < t \le t_{x1} \\ 1 & \text{for } t > t_{x1} \end{cases} \tag{5}$$

If the last interrupt was a random interrupt then the cumulative distribution functions of time interval to next interrupt are given as follows:

CDF of the interval t_{x2} from a random interrupt to the next timer interrupt

$$F_{x2}(t) = \begin{cases} 0 & \text{for } t \le 0 \\ t\, f_x & \text{for } 0 < t \le t_{x1} \\ 1 & \text{for } t > t_{x1} \end{cases} \tag{6}$$

There is no synchronization between timer and random interrupts so it's not known when the last timer interrupt was generated and when to expect the next. Due to periodic nature of timer interrupts, the next timer interrupt must appear within $1/f_x$ time. The probability is equally spread in $(0, 1/f_x)$.

CDF of the time t_{y2} from a random interrupt to the next random interrupt:

$$F_{y2}(t) = 1(t)(1 - e^{-\lambda t}) \tag{7}$$

Distribution function of time interval from a random interrupt to any interrupt (whichever comes first) is given the equation:

$$F_2(t) = 1 - (1 - F_{x2}(t))(1 - F_{y2}(t)) \tag{8}$$

$$F_2(t) = \begin{cases} 0 & \text{for } t \le 0 \\ 1 - e^{-\lambda t} + t f_x e^{-\lambda t} & \text{for } 0 < t \le t_{x1} \\ 1 & \text{for } t > t_{x1} \end{cases} \tag{9}$$

2.2 Cumulative distribution function

The Bayes equation for total probability is used to get the cumulative distribution function as a weighted sum of distribution functions of previously shown alternative cases.

$$F(t) = p_1 F_1(t) + p_2 F_2(t) = \frac{f_x}{f_x + \lambda} F_1(t) + \frac{\lambda}{f_x + \lambda} F_2(t) \qquad (10)$$

After simplification and grouping of variables:

$$F(t) = \begin{cases} 0 & \text{for } t \le 0 \\ 1 - e^{-\lambda t} + \frac{f_x \lambda}{f_x + \lambda} t e^{-\lambda t} & \text{for } 0 < t \le t_{x1} \\ 1 & \text{for } t > t_{x1} \end{cases} \qquad (11)$$

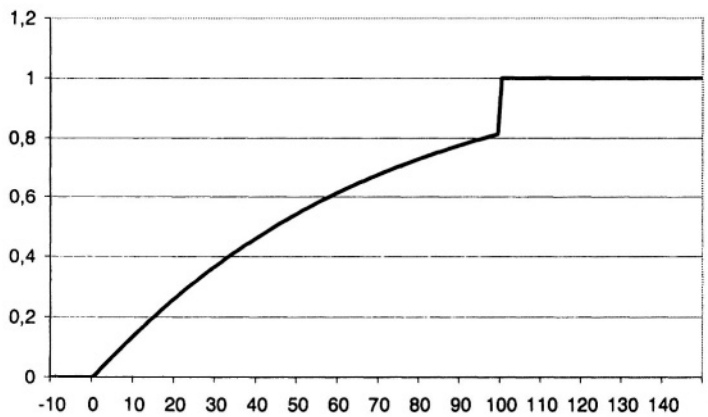

Figure 2. Example distribution function for $f_x = 0.01$ and $\lambda = 0.01$

The cumulative distribution function has following properties:

- Equals zero for negative time values
- Equals one for $t > 1/f_x$
- Is continuous excluding $t = 1/f_x$
- Difference of left and right side limits at $t = 1/f_x$ equals $f_x/(f_x+\lambda)\exp(-\lambda/f_x)$
- Is differentiable in all range excluding $t = 0$ and $t = 1/f_x$
- Is integrable in all its range - it consist of sum $const + e^t + te^t$

2.3 Density and intensity

The first derivative of the distribution function (density function)

$$f(t) = \begin{cases} 0 & \text{for } t \le 0 \\ \left(\frac{2f_x\lambda + \lambda^2}{f_x + \lambda}\right)e^{-\lambda t} + \frac{f_x\lambda^2}{f_x + \lambda}\left(te^{-\lambda t}\right) & \text{for } 0 < t < t_{x1} \\ \frac{f_x}{f_x + \lambda}\exp\left(\frac{\lambda}{f_x}\right)\,\delta(t - t_{x1}) & \text{for } t = t_{x1} \\ 0 & \text{for } t > t_{x1} \end{cases} \qquad (12)$$

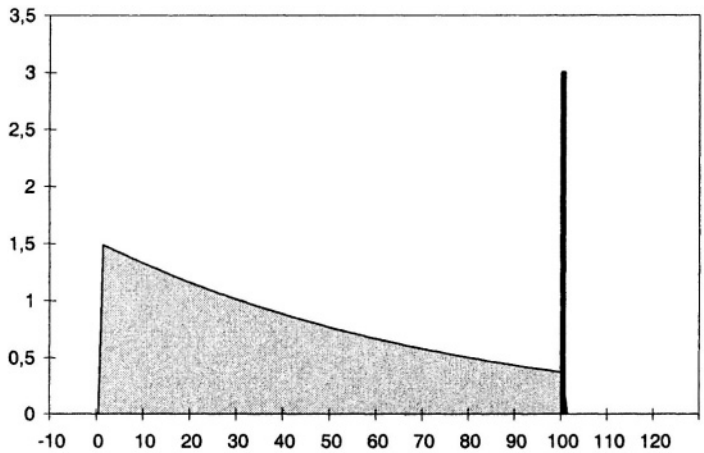

Figure 3. Example density function for $f_x = 0.01$ and λ=0.01

Density function of distribution of time interval between interrupts:

- Produces positive values in range $(0, 1/f_x)$
- Has Dirac's delta at $t=1/f_x$
- Is descending in range $(0, 1/f_x)$, slower than exponential distribution

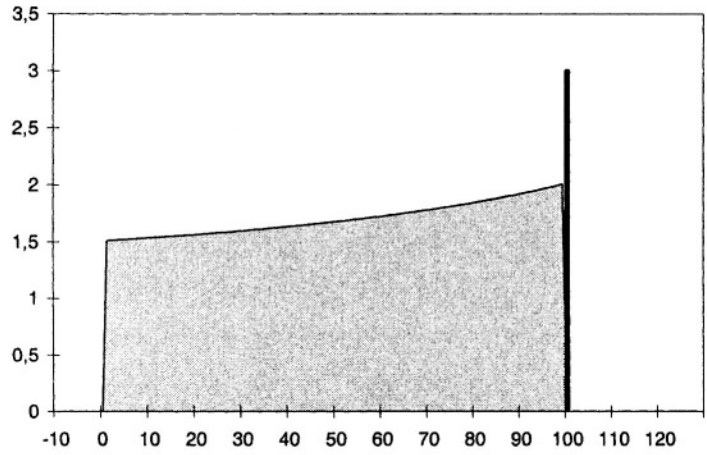

Figure 4. Example intensity function for $f_x = 0.01$ and $\lambda=0.01$

The intensity function of interrupt requests has following properties:

- Produces positive values in range $(0, 1/f_x)$
- Has Dirac's delta at $t=1/f_x$
- Is ascending in range $(0, 1/f_x)$
- If $\lambda>>f_x$ is similar to intensity of exponential distribution
- If $\lambda<<f_x$ produces small values for $t=(0, 1/f_x)$, Dirac's delta is dominating

3. EMPIRICAL RESULTS

Observations of moments when interrupts are requested have been taken to construct empirical distribution of time between two consecutive requests. Fast spin-lock procedure was used as a measuring routine and good approximation of CPU utilization by pooling driver or user multimedia application. To get reliable results 10^{10} measurements have been taken during each experiment. In most experiments $10^5 - 10^6$ request were detected. Density function of time interval between two consecutive interrupts has been plotted because it graphically shows better the nature of distribution than the cumulative distribution function. Having so big collection of empirical data, very high level of confidence was expected during statistical tests. Unfortunately majority of statistical analysis applications are not capable of proper handling of such large sets of data.

The histogram (density function) for Pentium III – 450MHz system working under Windows 98 is presented on figure 5. Nearly 700 thousands intervals has been measured and shown giving good approximation of distri-

bution shape. Horizontal axis represents length of time interval expressed in processor clock periods (450MHz). Vertical axis represents the number of samples that fit in range *(t, t+dt)*. Density function has not been scaled to have integral equal one. Empirical results are plotted as black line. Additionally light-gray line of theoretical histogram has been plotted. In the left part of the figure they fit almost perfectly, while the Dirac's delta is left hand diffused. Due to $f_x > \lambda$ relation, the descending nature of density function is weak, although can be observed.

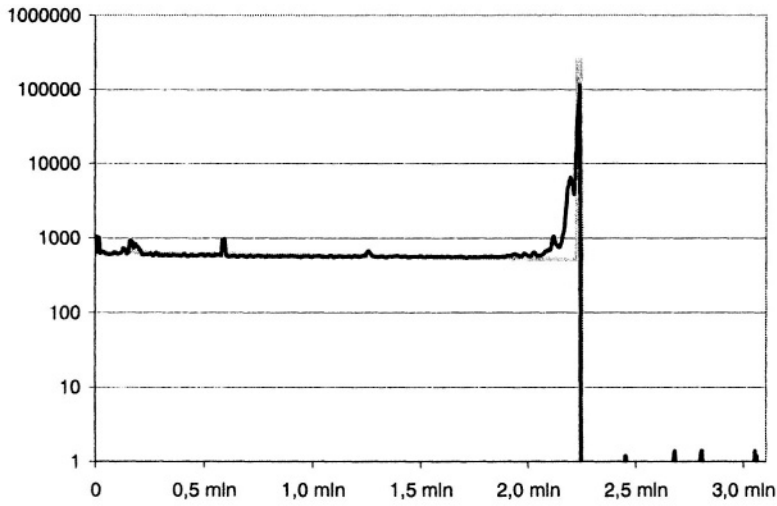

Figure 5. Empirical density function for $f_x = 200\text{Hz}$ and $\lambda \approx 81$

Time interval between two consecutive interrupts seen by user-level application may significantly differ from the real one. The histogram (density function) of time interval between interrupts for Pentium 4 system 2,5GHz working under Win-XP system is presented on figure 6. Three interrupt requests of periodic nature can be observed in this system. Additionally a small peak at 78 million cycles occurs. The reason for odd shape of histogram is that a measuring procedure runs as a normal user-level process in highly over-loaded single-processor system. When CPU is switched to another process the spin-lock loop used for measurement can not detect interrupts since it is not running. In fact there is only one periodic interrupt in presented system - the first peak in histogram at $\tau = 1.33\text{ms}$. The second peak occurring at 4τ the third at 12τ and the fourth at 24τ are caused by scheduler that stopped measuring procedure giving other programs 4, 12, or 24 "chunks" of CPU time. Since processor's clock count register (used for

measurement) was incremented while spin-lock procedure was sleeping the application after wake-up observed it as a single, very long interrupt.

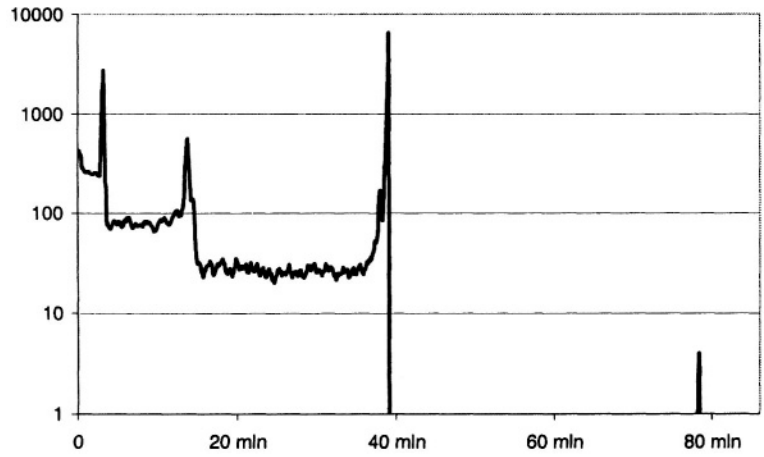

Figure 6. Empirical density function from application point of view

4. CONCLUSION

Statistical analysis of interrupt request behavior can be a helpful tool in design and validation of systems with coexisting interrupt driven and pooling I/O service routines. Additionally the influence of interrupt requests on application execution is significant in soft time critical programs such as audio or video recording and playback. Presented simple model of delays caused by operation system can be effective means of describing execution platform from the applications point of view without going into details of hardware and system software. Statistical description of machine and system influence on application is far less exact than behavioral model of operating system, but also far more simple and easy to apply. Model of interrupts slowing down application execution can be regarded as formal method of defining "smoothness" of application execution. Presented model is especially suited to low-end audio and video systems rapidly growing in popularity.

5. REFERENCES

[1] L. Abeni, G. Lipari, "Compensating for interrupt process times in real-time multimedia systems", *Third Real-Time Linux Workshop,* Milano 2001.

[2] L. Abeni, "Coping with interrupt execution time in real-time kernels: a non intrusive approach", *Proceedings of the IEEE Real-Time Systems Symposium WIP,* London 2001.

[3] R. Nelson, *Probability, Stochastic Processes and Queuing Theory – The Mathematics of Computer Performance Modeling,* Springer Verlag, New York 1995

[4] W. Noworyta – "Universal Model of Interval between Interrupt Requests" – to be published

[5] M. Gardner, J. Liu, "Performance of Algorithms for Scheduling Real-Time Systems with Overrun and Overload", *IEEE Proceedings 11th Euromicro Conference Real-Time Systems,* June 1999.

[6] Wolfgang A. Halang, "Contemporary research on real-time scheduling considered obsolete", *27th IFAC/IFIP/IEEE Workshop on Real-Time Programming,* May 14-17 2003 Lagow, Poland.

[7] B. Srinivasan, S. Pather, R. Hill, F. Ansari, D. Niehaus, "A firm real-time system implementation using commercial off-the-shelf hardware and free software", *Proceedings of the IEEE Real-Time Technology and Applications Symposium,* 1998.

[8] G. Bernat, A. Burns, A. Llamosi, "Weakly Hard Real-Time Systems", *IEEE Transactions on Computers* Vol. 50 No. 4 April 2001