Burns Standard Notation for real time scheduling

Robert I. Davis

University of York
rob.davis@york.ac.uk

1 Introduction

During the last 20 years, there has been a significant growth in the number of people active in the real-time scheduling community and consequently a large increase in the number of research publications. Today, more than ever it is important that we make our latest research (whether it is a short workshop abstract like this one or a lengthy journal paper) as easy as possible for others to understand and build upon.

As readers and reviewers of papers on real-time scheduling, we have all experienced the difficulty and at times hair-pulling frustration involved in trying to decipher complex analysis embodied in numerous equations using unfamiliar, arbitrary and sometimes downright cryptic notation [1]. We have all also experienced the pleasure of reading interesting, insightful, well-structured papers with clear step-by-step analysis, that uses precise terminology, and a concise, consistent and well thought-out notation [13].

Through the volume and the quality of the research he has produced (450 publications and counting), the number of PhD students he has supervised and nurtured into independent researchers, and the number of people reading his work (approaching 15,000 citations), Alan Burns continues to have a substantial and guiding influence on the de-facto standard terminology and notation that has been adopted by many in the real-time scheduling community. We do not claim that he invented this notation, but that over the years he has been instrumental in extending and shaping it into a form suitable for continued use.

In recognition of $Alan's$ $0x3C$ birthday[1] and his enduring contribution to the field of real-time scheduling, we hope that this notation, summarised below, will from hereon be referred to as ***Burns Standard Notation***. Our aim is for it to be used in future by all attending the workshop or reading this abstract, and thus become so widespread and standardised upon that in a few years it will be hard to find a new paper on real-time scheduling that does not make use of it.

2 Guidelines

Burns Standard Notation describes the properties of a real-time system and in particular the set (τ) of n tasks that execute on it, where each task τ_i is identified by a unique index i from 1 to n. This notation evolved over a number of years according to a set of informal

[1] Beyond a certain age one should think in hexadecimal and still claim to be thirty something. At the time of writing, the author had recently turned thirty.

guidelines. These guidelines provide both a rationale for the choices made and enable extension of the notation to new properties or parameters in a consistent way.

The guidelines used in creating and extending Burns Standard Notation are as follows:

(i) Subscripts refer to a task index, for example D_i, with a second subscript (if required) referring to the index of a specific invocation or job of that task, for example $d_{i,j}$.
(ii) Upper case letters are used for offline properties that are relative rather than absolute, for example D_i is the relative deadline of task τ_i.
(iii) Lower case letters are used for online properties that are absolute, for example $d_{i,k}$ is the absolute deadline of the kth job of task τ_i.
(iv) Functions are used for properties that vary with respect to some parameter (often time or a time interval). For example $c_i(t)$ denotes the remaining execution time of task τ_i at time t, and $I_i(t)$ denotes the maximum interference from task τ_i in an interval of length t.
(v) Superscripts are used to qualify different variants of a property, for example R_i^{LO} denotes the response time of task τ_i when the system is in a low criticality mode.
(vi) Sets of tasks are described as functions of the task index, for example $lp(i)$ is the set of tasks with priorities lower than that of task τ_i.

3 Burns Standard Notation

Burns Standard Notation is given in the table below, in alphabetical order. Our aim is for this notation to become the accepted standard for real-time scheduling papers.

Notation	Terminology	Meaning
B_i	Blocking	The longest time for which task τ_i can be prevented from executing by tasks of lower priority.
C_i $c_i(t)$	Worst-case execution time	An upper bound on the longest possible execution time of task τ_i. Worst-case remaining execution time of task τ_i at time t.
D_i	Relative Deadline	The longest elapsed time that task τ_i is permitted to take from being released until it completes execution.
$E(t)$	Error recovery overhead	The total time spent recovering from errors in a time interval of length t.
F_i	Final non-pre-emptive region	The maximum length of the final non-pre-emptive region of task τ_i.
H	Hyperperiod	The Least Common Multiple of all task periods.
$hp(i)$ $hep(i)$	Set of higher priority tasks	$hp(i)$ is the set of tasks with higher priority than task τ_i, whereas $hep(i)$ is the set of tasks with higher or equal priority to task τ_i.

Notation	Terminology	Meaning
J_i J_k	Release Jitter Job k	The longest possible time from the notional arrival of a job of task τ_i until it is released i.e. becomes ready to execute. In the case of papers discussing systems of independent jobs, then J_k is used to mean the job with index k.
L_i	Criticality level	In a mixed criticality system, the criticality level of task τ_i.
$lp(i)$ $lep(i)$	Set of lower priority tasks	$lp(i)$ is the set of tasks with lower priority than task τ_i, whereas $lep(i)$ is the set of tasks with lower or equal priority to task τ_i.
m		Number of processors
n		Number of tasks
P_i	Priority	The priority of task τ_i, used to determine which of a competing set of ready tasks should be executed. Where it is possible to do so without loss of generality, the priority of a task is often assumed to equate to its index i.
R_i	Worst-case response time	The longest possible elapsed time from the release of any job of task τ_i until the completion of that job.
S_i	Slack	The maximum amount of additional interference that task τ_i may be subject to without missing a deadline.
s	Speed	The speed of the processor.
T_i	Period	The minimum inter-arrival time between jobs of task τ_i.
U_i	Utilisation	The processor utilisation of task τ_i, $U_i = C_i/T_i$. (U is the utilisation of the task set τ).
τ_i	Task	The task with index i.
τ	Task set	
W_i or w_i	Window or busy period	The length of a priority level i busy period or scheduling window. Used in schedulability analysis equations where the response time is not computed directly.

The symbols C_i, T_i, τ_i and U were used in the famous paper by Liu and Layland [10] in 1973. Many of the other symbols, including B_i, D_i, I_i, J_i, R_i, and $hp(i)$ became established following their use in one of Alan Burns seminal contributions to the analysis of fixed priority scheduling [2]. Other symbols were introduced in subsequent papers in the mid 1990s (F_i, P_i and superscripts [5], S_i [6], $E(t)$ [11]), whereas L_i [4] is a more recent addition.

References

1. It would be unfair to pick out any one paper for criticism but I see you checked hoping it wasn't yours! A.
2. N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284 –292, 1993.
3. Neil Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

4. S.K. Baruah, A. Burns, and R.I. Davis. Response-time analysis for mixed criticality systems. In *proceedings Real-Time Systems Symposium*, pages 34 –43, 29 2011-dec. 2 2011.
5. Alan Burns. Preemptive priority-based scheduling: An appropriate engineering approach. In *Advances in Real-Time Systems, chapter 10*, pages 225–248. Prentice Hall, 1994.
6. R.I. Davis, K.W. Tindell, and A. Burns. Scheduling slack time in fixed priority pre-emptive systems. In *proceedings Real-Time Systems Symposium*, pages 222 –231, 1993.
7. T. Ebenlendr, M. Krcal, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, pages 1–19, 2012. (Extended abstract published in the SODA 2008 Proceedings).
8. Edward A. Lee, Stephen Neuendorffer, and Michael J. Wirthlin. Actor-oriented design of embedded hardware and software systems. *Journal of Circuits, Systems, and Computers*, 2, 2003.
9. J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
10. C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, January 1973.
11. K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (can) message response times. *Control Engineering Practice*, 3:1163–1169, 1995.
12. José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. In *ESA*, pages 530–542, 2011.
13. It would be unfair to pick out any one paper for praise but I see you checked this time hoping it was yours! Z.