

Using PERTS and PERTS*Sim to Analyze End-to-End Completion Times

Ben Watson, watson@tripac.com, Tri-Pacific Software, Alameda, CA

Introduction

Real time systems can be characterized as a set of preemptible tasks, each task having a priority, an amount of work and a deadline. The *completion time* of a task is the elapsed time for a task to accomplish its work. A system is *schedulable* when each task in the system accomplishes its work prior to its deadline. In other words, a system is schedulable when the completion time of every task in the system is prior to the task's deadline.

Rate Monotonic Analysis (RMA) provides the theoretical basis for a static schedulability model for a real time system. PERTS (Prototyping Environment for Real Time Systems) is a software tool that incorporates RMA schedulability models for a variety of system architectures. Using PERTS, a designer can confidently architect real time systems that are guaranteed to be schedulable under worst case conditions.

This paper addresses a particular class of real time systems whose tasks are composed of a set of sequentially executed sub-tasks. In such systems, the sub-tasks are characterized by an amount of work to accomplish, a fixed priority and a deadline, and are scheduled independently. When speaking of tasks with component sub-tasks, we are interested in the *end-to-end* completion time of the sequence of sub-task components. The end-to-end completion time is the elapsed time from the start of the first sub-task in the sequence until the completion time of the last sub-task. The schedulability criteria becomes that the end-to-end completion time of the task is less than its deadline.

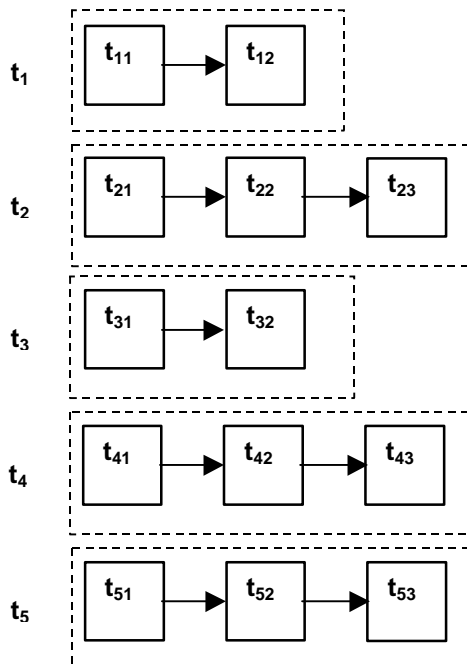


Figure 1 Sample Problem Structure

PERTS implements three different schedulability models for end-to-end completion analysis. The differences among the models are the assumed system architecture and task behavior. **HKL** and **SGL** are

limited to single processor architectures. **HKL** is applied to systems of periodic tasks while **SGL** addresses non-periodic tasks with variable release times and variable execution times. **End-to-End** analysis is applied to multiprocessor systems. Each model is described in more detail below. This paper presents a more formal definition of the systems under consideration in the next section. The next three sections present examples of using PERTS with each model. The final section identifies future directions for PERTS development.

Problem Statement

The problems that we will examine all have a similar structure, although there are differences in some of the details for each model. Figure 1 shows the general structure of the problems. The system consists of multiple tasks t_1, t_2, t_3, t_4 , each of which is composed of sub-tasks. The sub-tasks are identified as t_{ij} where i is the parent task number and j is the sequential occurrence of the sub-task within the sub-task chain. Each task t_i has a worst-case computation requirement C_i , a period T_i , and a deadline D_i . The computation requirement must be completed before the deadline. Correspondingly, each sub-task t_{ij} is characterized by the parameters C_{ij} , D_{ij} , and a fixed priority P_{ij} . In general, $0 \leq D_{i1} \leq D_{i2} \dots \leq D_{im} = D_i$, and $C_i = C_{i1} + C_{i2} + \dots + C_{im}$. That is, the sub-task deadlines are all within the task deadline and the execution requirement for the task is the sum of the sub-task execution requirements. The entire system is assumed to be scheduled by priority, meaning that any sub-task that is executing can be preempted by another sub-task of a higher priority. Table 1 presents the parameter values for the example. P in this example is the *rate monotonic* priority for each task.

| Task | Sub-Task | T | C | D | P |
|-------|----------|-----|----|-----|-----|
| t_1 | | 40 | 6 | 40 | 40 |
| | t_{11} | 40 | 1 | 40 | 40 |
| | t_{12} | 40 | 5 | 40 | 40 |
| t_2 | | 100 | 20 | 100 | 100 |
| | t_{21} | 100 | 10 | 70 | 100 |
| | t_{22} | 100 | 5 | 100 | 100 |
| | t_{23} | 100 | 5 | 100 | 100 |
| t_3 | | 50 | 20 | 50 | 50 |
| | t_{31} | 50 | 8 | 50 | 50 |
| | t_{32} | 50 | 12 | 50 | 50 |
| t_4 | | 200 | 33 | 200 | 200 |
| | t_{41} | 200 | 10 | 200 | 200 |
| | t_{42} | 200 | 20 | 200 | 200 |
| | t_{43} | 200 | 3 | 200 | 200 |
| t_5 | | 400 | 24 | 400 | 400 |
| | t_{51} | 400 | 2 | 400 | 400 |
| | t_{52} | 400 | 12 | 400 | 400 |
| | t_{53} | 400 | 10 | 400 | 400 |

Table 1 Sample Problem Values

End-to-End Analysis

The End-to-End analysis model in PERTS follows closely the work done by Bettati and Liu [Bettati and Liu, 1992]. The model is designed for multiple processors although it can be applied to single processor systems. On single processor systems, the other models give better bounds on the completion times. That will be addressed later in the discussions of HKL and SGL. One of the issues in analyzing sequential chains of sub-tasks is synchronizing the sequential execution. Bettati proposed a method whereby each sub-task in a chain is released for execution at the worst-case completion time of its predecessor. For worst-case phasing, all of the sub-tasks are ready to execute at the same instant. The first sub-task in a

chain is release immediately. The other sub-tasks are delayed to allow the preceding sub-task to finish its work.

To demonstrate end-to-end analysis, we will assign the individual sub-tasks to one of three processors. Figure 2 shows the sub-task assignments to processors.

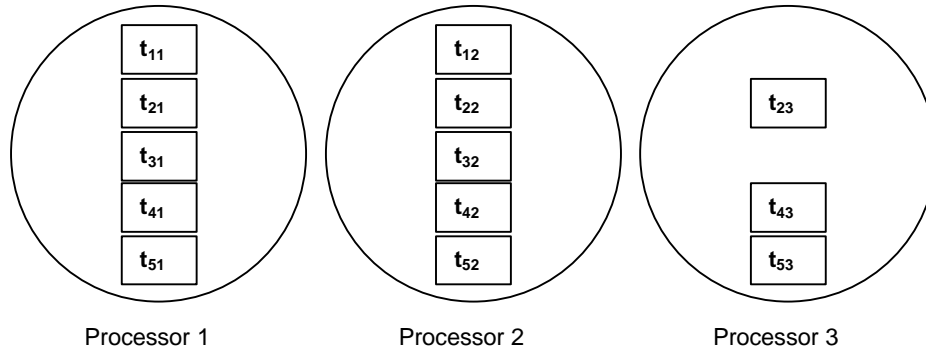


Figure 2 Sub-task processor assignments

Table 2 shows the results of running PERTS End-to-End analysis. Notice the cascade of the ready time down the chain of sub-tasks. Notice also that each task meets its deadline and thus the entire system is schedulable.

| Task | Sub-Task | C | Ready | Worst-Case Completion |
|----------------|-----------------|----|-------|-----------------------|
| t ₁ | | 6 | | 40 |
| | t ₁₁ | 1 | 0 | 1 |
| | t ₁₂ | 5 | 1 | 6 |
| t ₂ | | 20 | | 100 |
| | t ₂₁ | 10 | 0 | 19 |
| | t ₂₂ | 5 | 19 | 41 |
| | t ₂₃ | 5 | 41 | 46 |
| t ₃ | | 20 | | 50 |
| | t ₃₁ | 8 | 0 | 9 |
| | t ₃₂ | 12 | 9 | 26 |
| t ₄ | | 33 | | 200 |
| | t ₄₁ | 10 | 0 | 29 |
| | t ₄₂ | 20 | 29 | 76 |
| | t ₄₃ | 3 | 76 | 84 |
| t ₅ | | 24 | | 400 |
| | t ₅₁ | 2 | 0 | 31 |
| | t ₅₂ | 12 | 31 | 102 |
| | t ₅₃ | 10 | 102 | 120 |

Table 2 Multiprocessor End-to-End Results

This is a very nice result for the multiprocessor system that we have constructed. However, when we attempt this same analysis on a single processor, the results are not so nice. Table 3 shows the partial results. The worst-case completion times are computed using the standard time demand analysis for a single processor system. Obviously, the downstream sub-tasks have no time to perform their work.

| Task | Sub-Task | C | Ready | Worst-Case Completion |
|----------------|-----------------|----|-------|-----------------------|
| t ₁ | | 6 | | 40 |
| | t ₁₁ | 1 | 0 | 6 |
| | t ₁₂ | 5 | 6 | 11 |
| t ₂ | | 20 | | 100 |
| | t ₂₁ | 10 | 0 | unknown |
| | t ₂₂ | 5 | | |
| | t ₂₃ | 5 | | |
| t ₃ | | 20 | | 50 |
| | t ₃₁ | 8 | 0 | 26 |
| | t ₃₂ | 12 | 26 | 44 |
| t ₄ | | 33 | | 200 |
| | t ₄₁ | 10 | 0 | 183 |
| | t ₄₂ | 20 | 183 | unknown |
| | t ₄₃ | 3 | | |
| t ₅ | | 24 | | 400 |
| | t ₅₁ | 2 | 0 | unknown |
| | t ₅₂ | 12 | | |
| | t ₅₃ | 10 | | |

Table 3 Single Processor End-to-End Results

HKL Analysis

HKL (Harbour, Klein, Lehoczky) analysis is the PERTS implementation of a paper by Harbour, Klein and Lehoczky [Harbour, Klein, Lehoczky, 1994]. The analysis addresses periodic tasks running on a single processor and cannot be extended to multiprocessor systems. Like the End-to-End model, HKL assumes that ready time is used to synchronize sequential execution within the sub-task chains. One of the difficulties with the end-to-end model is an overly pessimistic worst-case completion time. In some cases, sub-tasks from the same task chain are used to calculate preemption and blocking terms. Since that case cannot arise because only one sub-task from each task chain can be ready to execute at a time. HKL includes an algorithm for removing such cases from the calculations which yields a tighter bound on the completion times. The results of running HKL analysis on the sample problem are shown in Table 4. Compared to end-to-end analysis, this is a much more favorable result since a single processor can be used.

SGL Analysis

A recent paper [Sun, Gardner and Liu, 1997] extends a similar analysis to non-periodic real time systems. In these systems, tasks, decomposed into chains of sub-tasks, are initiated at arbitrary times but do not repeat on a periodic basis. SGL analysis is intended to analyze an arbitrary mix of tasks that can be active within a system; therefore, the periods of the sub-tasks (and tasks) are ignored. The goal of SGL analysis is to discover a set of priorities for the sub-tasks that produce a schedulable system. An SGL run is shown in Table 5 where the completion times are similar to the HKL results. Notice that the longer running tasks do not show any interference from the shorter running tasks.

| Task | Sub-Task | C | Ready | Worst-Case Completion |
|----------------|-----------------|----|-------|-----------------------|
| t ₁ | | 6 | | 40 |
| | t ₁₁ | 1 | 0 | 1 |
| | t ₁₂ | 5 | 1 | 6 |
| t ₂ | | 20 | | 100 |
| | t ₂₁ | 10 | 0 | 36 |
| | t ₂₂ | 5 | 26 | 47 |
| | t ₂₃ | 5 | 47 | 72 |
| t ₃ | | 20 | | 50 |
| | t ₃₁ | 8 | 0 | 14 |
| | t ₃₂ | 12 | 14 | 26 |
| t ₄ | | 33 | | 200 |
| | t ₄₁ | 10 | 0 | 88 |
| | t ₄₂ | 20 | 88 | 180 |
| | t ₄₃ | 3 | 180 | 183 |
| t ₅ | | 24 | | 400 |
| | t ₅₁ | 2 | 0 | 185 |
| | t ₅₂ | 12 | 185 | 197 |
| | t ₅₃ | 10 | 197 | 390 |

Table 4 HKL Results

| Task | Sub-Task | C | Ready | Worst-Case Completion |
|----------------|-----------------|----|-------|-----------------------|
| t ₁ | | 6 | | 40 |
| | t ₁₁ | 1 | 0 | 1 |
| | t ₁₂ | 5 | 1 | 6 |
| t ₂ | | 20 | | 100 |
| | t ₂₁ | 10 | 0 | 36 |
| | t ₂₂ | 5 | 36 | 41 |
| | t ₂₃ | 5 | 41 | 46 |
| t ₃ | | 20 | | 50 |
| | t ₃₁ | 8 | 0 | 14 |
| | t ₃₂ | 12 | 14 | 26 |
| t ₄ | | 33 | | 200 |
| | t ₄₁ | 10 | 0 | 56 |
| | t ₄₂ | 20 | 56 | 76 |
| | t ₄₃ | 3 | 76 | 79 |
| t ₅ | | 24 | | 400 |
| | t ₅₁ | 2 | 0 | 81 |
| | t ₅₂ | 12 | 81 | 93 |
| | t ₅₃ | 10 | 93 | 103 |

Table 5 SGL Results

Future Directions

Of the three models discussed, only end-to-end implements shared resources. Shared resources are modeled as critical sections in the (sub-)tasks. Contention for the resources is controlled by a selection of

resource contention protocols. The formulations for both HKL and SGL include blocking from resource contention, but those terms have not been incorporated in PERTS. There are also some recent papers on analyzing periodic multiprocessor system. One of these papers [Sun, 1997] explores a variety of sub-task synchronization protocols and priority assignment heuristics. The analysis models from that paper produce much tighter bounds than any of the models in this paper. Our goal at Tri-Pacific Software is to incorporate this new work in a timely fashion.

References

- R. Bettati and J.W.S. Liu, "End-to-End Scheduling to Meet Deadlines in Distributed Systems," *Proceedings of the 12th International Conference on Distributed Computing Systems*, Yokohama, Japan, June, 1992.
- M.G. Harbour, M.H. Klein, and J.P. Lehoczky, "Timing Analysis for Fixed-Priority Scheduling of Hard Real-Time Systems," *IEEE Transactions on Software Engineering*, vol. 20, no. 1, pp. 13-28, Jan. 1994.
- J. Sun, *Fixed-Priority End-to-End Scheduling in Distributed Real-Time Systems*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.
- J. Sun, M.K. Gardner, and J. Liu, "Bounding Completion Times of Jobs with Arbitrary Release Times, Variable Execution Times, and Resource Sharing," Unpublished work for the University of Illinois, Urbana-Champaign, 1997.