

Unified Modeling Framework and Comprehensive Offline Analysis For Quantifiable Adaptive Real-Time Systems

H.Shrikumar, Krithi Ramamritham
 University of Massachusetts, Amherst MA 01003
 {shri,krithi}@cs.umass.edu

Abstract

We propose an integrated analytical framework based on markov and non-markov stochastic models, for the design and analysis of complex adaptive hard real-time systems, which have a mixture of both static and dynamic aspects, and a combination of both hard real-time (guarantee oriented) and soft real-time (performance/reliability oriented) requirements. Such task characteristics are now common in collaborative robotics, involving teams of robots and distributed adaptive signal processing. A contribution of our framework is that, along-with task and environment characteristics, it also explicitly captures adaptive choice in real-time schedulers in a unified framework.

I. Introduction

In classic scheduling approaches such as Rate Monotonic Analysis (RMA) [1], assumptions of strict tasks periodicity and worst-case task-interleaving enable making unconditional guarantees relating to tasks meeting their deadlines at the cost of some under-utilization (eg., 69%). Alternatively, dynamic scheduling disciplines such as EDF can potentially achieve 100% resource utilization, but guarantees can only be on-line and conditional upon feasibility at run-time.

These scheduling approaches are generally mutually incompatible. In collaborative robotics, the unpredictable environment and scarce resources suggest dynamically adaptive on-line scheduling. At the same time, off-line assurances (for eg., mission success rate) naturally calls for off-line analysis.

Further, the above schedulability analyses constitute an analytical framework distinct from that used to provide assurance of reliability metrics. Therefore, direct tradeoffs between schedulability properties (eg., guarantee ratios) and reliability metrics (eg., mean time to failure) can only be done through cumbersome design iterations.

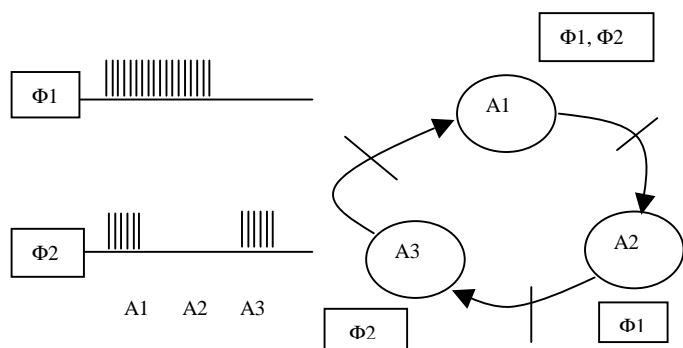


Fig 1. Task Bursts in a Single Robot Activity

Our integrated approach reduces the hard as well as soft real-time aspects of the tasks, as well as the execution environment, into a unified hierarchical stochastic model. This model can also capture the effect of potentially adaptive switching of different

scheduling disciplines at run-time. As an additional contribution, the analytical framework generates data regarding these subtle task-task-scheduler interactions, which can be captured off-line in a “Composability Table”. This lookup table of “sentinels” (described later) enables fast, safe and stable run-time adaptation.

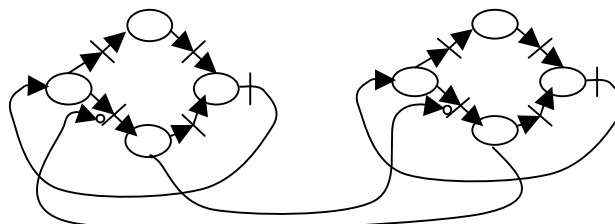


Fig 2. Task Interaction: Resource Contention

Incidentally, the above model can be studied using analytic markov-chain techniques that show promise of being more precise; however such techniques face problems with combinatorially explosive state spaces and numerical stiffness issues, and also cannot deal with real-world requirements that are inherently non-markovian. We demonstrate the effective use of mature Deterministic Stochastic Petri Net (DSPN) solver package (SPNP) [3], which is better at addressing all the above-mentioned aspects of this problem in one unified and compositional framework.

II. “UMass Walker” : Robotics Task Set

Figure 1 shows a typical robotic task. This task has three states. Within each state (for example, A1), a set of strictly periodic tasks, PID motor-controller loops(Φ_1) and sensor loops(Φ_2), are enabled, in order to accomplish some goal, say, a movement or a grasp. The duration of each state is statistical and depends on the environment. Any activity of the robot consists of task bursts pertaining to the different states, one set following the other; the recurring cycle of these periodic task-bursts in any activity constitutes the *quasi-periodic characteristic* of the task set. Note that during the of activation of this task burst, other task bursts belonging to other unrelated activities in the same robot CPU, and in other robot CPUs, are also being initiated and terminated, independently, and at different times.

III. Task & Scheduler Interactions as a DSPN

As seen above, the robot task can be readily represented as a Petri Net. By allowing the representation to be a DSPN, we show below how we can further capture other system semantics.

Resource Contention: Fig 2 shows how a DSPN can capture simple blocking. The two bottom most states of the two robotic activities conflict on the same resource; this is captured in the form of two “inhibit transitions” in the DSPN.

Scheduler: Likewise, a set of adaptively chosen real-time schedulers themselves can be represented in the DSPN, as a CPU-contention. For example, an RMA scheduler can be represented as a single DSPN place, having a single-token at initialization time. A set of prioritized DSPN transitions from

this single place that “un-inhibits” every other transition in turn according to static priority represents the dispatching component of the scheduler. (figure not shown due to space constraints).

Adaptive Sentinels: Adaptive scheduling is used to handle transient overload, and has performability implications. It is implemented in an adaptive real-time system by searching through the composite state-space of all tasks for states with a CPU overload and protecting them from occurring with “sentinels”. A sentinel is an OS monitor compiled into the executable, which actively prevents one of the conflicting tasks from making progress for the duration of the overload, to help reduce some workload. This is a good example of a performance/ real-time guarantee/reliability tradeoff.

In fig 3, each of these two tasks can cause an overload due to a certain shared resource. Each task can share this resource across its multiple instances (within type A or B), but not across different task types. That is to say that when any instance of task A is executing state A3, any instances of task B also be in state B3, or vice versa, represents an overload condition. A “sentinel” that prevents this is implemented using blocking/queuing. This is represented using holding states (A11 and B1) and inhibitor arcs (near A3a and B3a).

Reward Rates for QOS: Places in the DSPN are assigned Reward Rates based upon the value of each task to the user (quality of service). The resulting reward rate of the DSPN can be used to analyze transient as well as steady-state performance of a system (QOS parameters), or its reliability and mean time to failure properties (Reliability).

Reward Rates for Reliability: It is easy to see that a similar DSPN model can also be constructed to compute statistical reliability and MTTF (mean time to failure) rates for a system. This is a straightforward extension of this work, and we do not address it in this paper for reasons of space.

IV. Experimental Results, Conclusion

As a truthfully realistic basis for the experiments presented in this paper, we have constructed a quasi-periodic task set whose task parameters are based on real robotic “Walker” developed at the University of Massachusetts[2].

Fig 4 shows how the expected value of the normalized reward rate (z axis) varies with Task B period and number of instances of Task B. Unsurprisingly, the normalized reward accumulated by high-reward task A drops more drastically when the number of the instances (left-side axis) or speed (right-side axis) of the competing but non-remunerative task B is larger. What is

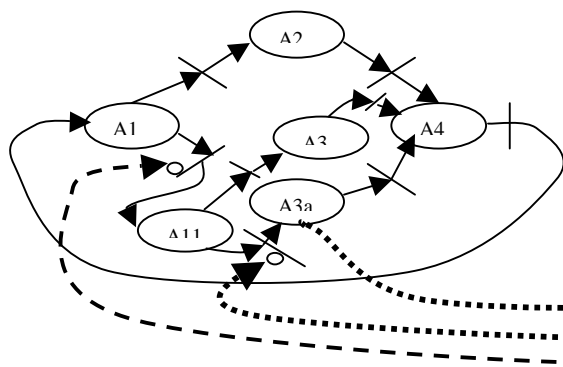


Fig 3. Scheduler Sentinels represented in DSPN

remarkable in Figure 4 is the wild oscillations of the expected value of the reward rate with even slight changes in the task B time constants. This suggests the occurrence of a phenomenon analogous to *entrainment*, defined in physics as the close temporal interaction of oscillators in the presence of non-linearities (shared resource), resulting in a lock-step rhythm, which is sometimes favorable and sometimes not.

Traditional hard real-time scheduling fail to take full advantage of this knowledge. Worst case analysis, as in RMA, typically operate on the outer envelope of these oscillations leading to an undocumented 2:1 underutilization over and above the known 69% RMA utilization bound. Simulation driven schedulability analysis, on the other hand, pick a random locus of some weighted average through these oscillations depending on the quirks in statistical generators that may not be well correlated with real-life workload – this risks run-time failures.

In the “Composability Approach” we propose here, we are able to perform an exact analysis. A Composability Table can now be built indexed by the following observables: TaskB instances and TaskB period. The output of this lookup table is the adaptive scheduling/allocation choice for this “sentinel”. This OS is now aware of the exact resource contention and can adaptively select sentinels for max-value with zero risk of causing overload.

References

- [1] L. Sha, R. Rajkumar, and J. P. Lehoczky. *Priority Inheritance Protocols: An Approach to Real-Time Synchronization*. IEEE Transactions on Computers, vol. 39, pp. 1175-1185, Sep. 1990.
- [2] Huber M, Grupen RA, *Learning to Coordinate Controllers - Reinforcement Learning on a Control Basis*, Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 1366-1371
- [3] J Muppala, S Woolet, K Trivedi, *Real-Time Systems Performance in the Presense of Failures*, IEEE Computer, Vol 24(5), May 1991.

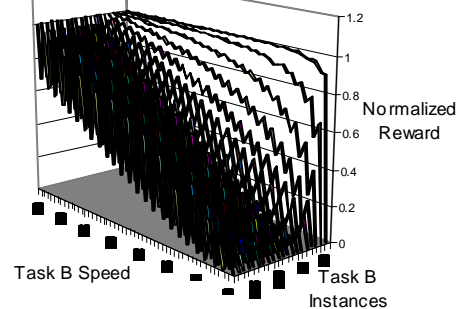


Fig 4. Normalized Task A Reward v/s Task B load-level

