Compositional Schedulability Analysis for Cyber-Physical Systems

Arvind Easwaran and Insup Lee
Department of CIS, University of Pennsylvania
PA, 19104, USA
{arvinde,lee}@cis.upenn.edu

Abstract

Cyber-physical systems (CPSs) are becoming all-pervasive, and due to increasing complexity they are designed using component-based approaches. Temporal constraints of such complex CPSs are then modeled using hierarchical scheduling frameworks. Therefore, there is a need to develop compositional schedulability analysis techniques for such CPSs. In this essay, we describe one such CPS present in air-crafts. We also discuss the hierarchical frameworks that are found in these systems, and highlight shortcomings of existing techniques in analyzing them.

Cyber physical systems. Embedded systems are widely applicable in today's world largely due to the ever decreasing cost of resources such as computing power and communication bandwidth. Of particular interest, are embedded systems that use the discrete and powerful world of computing to monitor and control the continuous dynamics of physical and engineered systems. These are known as *cyber-physical systems* or *deeply embedded systems*. There exist many examples of CPSs in our day-to-day lives such as medical devices, avionics systems and factory automation. It is desirable that these systems operate correctly, efficiently and in real-time.

Composition for CPSs. Typically, CPSs consist of a combination of different types of resources, programmable components, sensors, etc. Such re-configurable architectures are essential to adapt these systems to a variety of applications. With increasing complexity and large scale, there is a need for advanced design and analysis tools for CPSs. Component-based engineering, which involves compositional system modeling and analysis, is widely used for this purpose. It is founded on the paradigm that a complex system can be designed and analyzed by decomposing it into simpler components, and then composing the components using interfaces that abstract complexities.

CPSs interact with the physical world in many safety-critical domains like aviation, medicine, etc. Hence, they are subject to certification with respect to government specified regulations. In addition, these domains also have some form of timing requirements that must be satisfied by systems. Component-based real-time CPSs often involve hierarchical scheduling frameworks that support resource sharing among components under different scheduling algorithms. This framework can be represented as a tree of nodes, where each node denotes a component comprising of some real-time workload and a scheduling policy. To help with certification, it is then essential to develop schedulability analysis techniques for such hierarchical frameworks. Furthermore, to take advantage of component-based engineering, it is desirable to achieve this analysis *compositionally*, i.e., we should be able to check schedulability of the system by composing interfaces that abstract component-level resource demand.

ARINC-653 avionics RTOS. ARINC standards, developed and adopted by *Airlines Electronic Engineering Committee* (AEEC), deliver substantial benefits to airlines and aviation industry. In particular, the 600 series ARINC specifications and reports define enabling technologies that provide a design foundation for digital avionics systems. Within the 600 series, the ARINC specification 653-2 part I [2] (henceforth referred to as ARINC-653), defines a general-purpose Application/Executive (APEX) software interface between the operating system of an avionics computer and the application software.

As described in ARINC-653, among other things, the CPS in an aircraft comprises of one or more *core modules* connected with one another using switched Ethernet. Each core module in turn, consists of one or more processors, memory and network interfaces. An example partial model of this avionics system is illustrated in Figure 1, where it shows single redundancy. A core module supports time and space partitioned execution of one or more avionics applications. Each independently executing function is called a *partition* which in turn comprises of one or more real-time *processes*. As shown in Figure 2 (*cf.* Figure 1.2 in ARINC-653), the core module software architecture comprises of (1) application layer software including partitions and processes within partitions, (2) APEX (application executive) interface, and (3) an OS kernel. The APEX interface defines a set of API that applications can use to control their

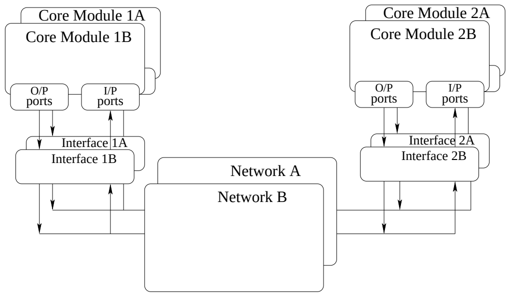

Figure 1. CPS in an aircraft

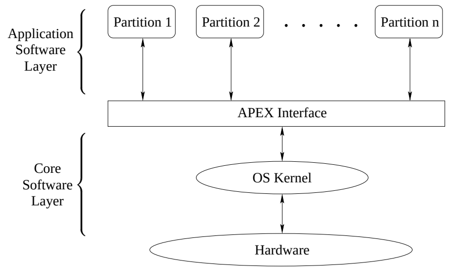

Figure 2. Core module software architecture

scheduling, communication or status information. The OS kernel supports a two-level hierarchical scheduling framework: a global partition-level scheduler and local process-level schedulers within partitions.

Processes within a partition can be aperiodic or periodic with pre-period deadlines, and can communicate with each other using input and output ports either asynchronously or synchronously. Sequences of such communicating processes form dependency chains, and designers can specify end-to-end latency bounds for them. Also, a process can be blocked by other lower priority processes in the same partition due to mutual exclusion requirements.

There are several problems related to the system described above, that must be addressed. For scheduling partitions, it is desirable to abstract the communication dependencies between processes in the form of process parameters like offsets, jitter and pre-period deadlines. This simplifies a global multi-partition scheduling problem, into several local single partition scheduling problems. The process deadlines must also guarantee satisfaction of end-to-end latency bounds specified by the designer. Given such processes, we must then generate scheduling parameters for the partitions. The resulting partition schedule must provide sufficient processor capacity to schedule processes within partitions. Furthermore, all these parameters must also account for blocking and preemption overheads incurred by processes and partitions.

Compositional schedulability analysis. Traditionally the partition scheduling problem described above has been addressed in an ad-hoc fashion, based on interactions between the system designer and vendors who provide the partitions. Although many different ARINC-653 platforms (*cf.* [1]) exist, there is a lack of techniques to automate this scheduling process. One such technique studied by us is compositional schedulability analysis using resource models [7, 5].

A resource model represents the characteristics of a resource supply, and hence can be used to abstract resource demand of components in their interfaces (ARINC-653 partitions can be regarded as components). A periodic resource model $\Gamma = (\Pi, \Theta)$, represents a resource supply that provides Θ units of resource in every Π time units. Similarly, an explicit deadline periodic resource model $\Omega = (\Pi, \Theta, \Delta)$, represents a resource supply that repetitively provides Θ units of resource in Δ time units, with period of repetition Π. In our analysis, we abstract component demands into interfaces using resource models, and then compose these interfaces to check schedulability of hierarchical frameworks. We have developed these techniques under deadline-monotonic and earliest deadline first schedulers.

Conclusion and challenges. Although many different resource model based compositional analysis techniques exist [7, 5, 3, 6, 4], none of them accurately account for preemption and blocking overheads. To efficiently analyze and schedule ARINC-653 partitions, we must then extend these techniques in this direction. Another important aspect is handling of inter- and intra-component communication dependencies. Some solutions have been proposed for compositional analysis in the presence of such dependencies [3, 6, 4]. However, a comprehensive theory addressing this problem is still to come. In particular, there is a need to abstract communication dependencies and end-to-end latency specifications in component interfaces. Thus in conclusion, we must extend existing techniques, as well as develop new techniques to analyze the schedulability of component-based real-time CPSs.

References

[1] Green Hills Software, ARINC 653 partition scheduler. In *www.ghs.com/products/safety_critical/arinc653.html*.

[2] Avionics application software standard interface: Required services (ARINC 653-2). Technical report, AEEC, 2006.

[3] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *EMSOFT*, 2004.

[4] R. I. Davis and A. Burns. Resource sharing in hierarchical fixed priority pre-emptive systems. In *RTSS*, 2006.

[5] A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using EDP resource models. In *RTSS*, 2007.

[6] S. Matic and T. A. Henzinger. Trading end-to-end latency for composability. In *RTSS*, 2005.

[7] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS*, 2003.