Networks of Real-Time Actors
Schedulability Analysis and Coordination

Mohammad Mahdi Jaghoori\textsuperscript{1,*}, Ólafur Hlynsson\textsuperscript{2}, and Marjan Sirjani\textsuperscript{2,3}

\textsuperscript{1} CWI, Amsterdam, The Netherlands \hspace{1cm} \textsuperscript{2} Reykjavik University, Iceland \hspace{1cm} \textsuperscript{3} University of Tehran, Iran

jaghoori@cwi.nl, \{olafurh05,marjan\}@ru.is

Abstract. We present an automata theoretic framework for modular schedulability analysis of networks of real-time asynchronous actors. In this paper, we use the coordination language Reo to structure the network of actors and as such provide an exogenous form of scheduling between actors to complement their internal scheduling. We explain how to avoid extra communication buffers during analysis in some common Reo connectors. We then consider communication delays between actors and analyze its effect on schedulability of the system. Furthermore, in order to have a uniform analysis platform, we show how to use \textsc{Uppaal} to combine Constraint Automata, the semantic model of Reo, with Timed Automata models of the actors. We can derive end-to-end deadlines, i.e., the deadline on a message from when it is sent until a reply is received.

1 Introduction

Schedulability analysis in a real-time system amounts to checking whether all tasks can be accomplished within the required deadlines. In a client-server perspective on distributed systems, tasks are created on a client, sent to the server (e.g., as a message), and then finally performed on the server. A deadline given by the client for a task covers three parts: the network delay until the message reaches the server, the queuing time until the task starts executing, and the execution time. In case a reply is sent back to the client, an end-to-end deadline also includes the network delay until the reply reaches the client and is processed.

In previous work \cite{10,15,17}, we employed automata theory to provide a modular approach to the schedulability analysis of real-time actor models, assuming direct and immediate communication between actors, i.e., zero communication delays. An actor \cite{1,13} (à la Rebeca \cite{20}) is an autonomous entity with a single thread of execution. Actors communicate by asynchronous message passing, i.e., incoming messages are buffered and the code for handling each message is defined in a corresponding method. We model each method as a timed automaton \cite{3} where a method can send messages while computation is abstracted in passage of time. In our framework, an actor can define a local scheduler and thus reduce

\* The work by this author is supported by the HATS project (EU FP7-231620).
the nondeterminism; a proper choice of a scheduling strategy is indeed necessary to make the actor schedulable.

Section 2 explains a modular way to analyze a system of actors. To be able to do so, the expected usage of each actor is specified in a separate timed automaton, called its behavioral interface; this is a contract between the actor and its environment [23], which among other things, includes the schedulability requirements for the actor in terms of deadlines. Every actor is checked individually for schedulability with regard to its behavioral interface. We showed in [16] that schedulable actors need finite buffers; the upper-bound on buffer size can be computed statically. When composing a number of individually schedulable actors, the global schedulability of the system can be concluded from the compatibility of the actors [17]. Being subject to state-space explosion, we gave a technique in [17] to test compatibility.

The contribution of this paper is twofold. First in Section 3, we extend the above framework with Reo [4] to enable exogenous coordination of the actors. This provides a separation of concerns between computation and coordination. Reo can be used as a “glue code” language for compositionally building connectors that orchestrate the cooperation between components or services in a component-based system or a service-oriented application. An important feature of Reo is that it allows for anonymous communication, i.e., the sender of a message does not need to know the recipient; instead the Reo connector will forward the message to the proper receiver.

With Reo, individually schedulable actors can be used as off-the-shelf modules in a wider variety of network structures. This requires a new compatibility check for our analysis that incorporates the Reo connectors. Our extension preserves the asynchronous nature of the actors, therefore the Reo connectors must have a buffer at every input/output node, which may lead to state-space explosion. To avoid this problem, we provide techniques to optimize the analysis by reusing internal actor buffers in the Reo connectors that are single-input and/or single-output. We show that in this approach the upper-bound on the size of the buffers of the schedulable actors need not be increased. In Section 5, we give examples of other Reo connectors that can take advantage of the same optimization technique. In any case, we assume coordination and data flow by Reo happens in zero time.

As our second contribution, we analyze in Section 4 the effect of communication delays on the schedulability of a distributed system. For simplicity in presentation, we assume no coordination with Reo in this section. The communication medium between every pair of actors is modeled abstractly by a fixed delay value, called their distance. We first describe how to implement the effect of delay on messages in an efficient manner with respect to schedulability analysis. Secondly we extend the compatibility check to take message delays into account. The latter is non-trivial because sending and receiving messages do not happen at the same time any more. Nevertheless, this complication can be hidden from the end user by implementing it in an automatic test-case generation algorithm.