Chapter 43
A Decentralized Control Architecture to Achieve Synchronized Task Behaviors in Autonomous Cooperative Multi-Robot Systems

Gen'ichi Yasuda
Nagasaki Institute of Applied Science, Japan

ABSTRACT

This chapter describes a method for designing decentralized simulation and control architecture for multiple robot systems based on the discrete event net models. Extended Petri nets are adopted as an effective tool to describe, design, and control cooperative behavior of multiple robots based on asynchronous, concurrent processes. By hierarchical decomposition of the net model of the overall system, global and local Petri net models are assigned to the upper level and the lower level controllers, respectively. For the lower level control, individual net models of robots are executed on separate local controllers. The unified net representation for cooperative control is also proposed. Overall control software is implemented and executed on a general hierarchical and distributed control architecture corresponding to the hardware structure of multiple robot systems.

INTRODUCTION

As a wide range of applications emerges in various domains of robotics, multiple robot systems with autonomous cooperative behavior are gaining increasing importance (Lepuschitz et al., 2011; Kantaros & Zavlanos, 2016). A multi-robot system should be realized to achieve an advanced functionality that cannot be realized by a single robot through some kind of cooperation, such as mutual handling of the partner robot or shared object, and map exploration in large-scale fields (Zhou et al., 2013). The key solution for multi-robot systems is to realize the cooperation by autonomous robots, which is different

DOI: 10.4018/978-1-5225-8060-7.ch043

from generic centralized control systems. The control architecture should be intrinsically non-centralized, parallel control architecture. So, the modeling and analysis of multi-robot systems always must meet with difficulties related to the cooperation problem under non-centralized control. In addition to existing frameworks for embedded system design (Gargantini et al., 2009), integrated design environments should support the entire design process from specification to implementation, considering typical features of robotic activities apart from the conventional paradigms for control.

For robots to work together efficiently and reliably to accomplish tasks that are intrinsically distributed in space, time, or functionality, it requires increasing cost and system complexity in control and communication. Currently, every robot carries its own controller and their actions are coordinated by telecommunication among the robots, based on serial architecture just like trees. Autonomous robots are able to cooperate and partition the global task, and then independently plan and execute their own tasks. However, because of time uncertainty, it makes the scheduling problem of the coordinator non-trivial. From a hardware-oriented point of view, a robot itself is composed of modularized components for different functions, such as vehicle, manipulator, end-effector, hand-eye camera, etc., which are computationally independent from each other and are controlled in parallel; independent sub-controller for each component. Current robot sub-controllers are mostly dedicated to centralized motion control at this component level (Melkou & Hamerlain, 2014; Mnasser et al., 2014; Mousa et al., 2015; Rajasekaran et al., 2014), where the communication between the sub-controllers can make cooperation between the components possible through a high-speed bus architecture or point-to-point architecture. Multi-robot systems with a centralized controller can be easily implemented in a well-defined environment, but often perform rather poorly under unknown dynamic environments. The application of model-based techniques is not suitable, because the growing complexity makes the maintenance of a model of the system dynamics unfeasible. Generally, complex control systems are partitioned in multiple components for simpler programming and management, but interrelationships become more complex. If the computer hardware architecture reflects the robot hardware architecture, a clear overview of the system can be provided, making it easier to develop, to debug and to extend. One of the advantages is that each processor can be specialized to its own job and the programmer can concentrate on the control algorithms separated from the input/output functions. Thus, distributed methods are more attractive due to their robustness, flexibility, and adaptability. Although, by allocating one processor to each component, the components can work in parallel and the execution time is reduced, an integral controller with parallel architecture should be implemented to perform the complex coordination tasks for integrated management of multi-robot task execution in lots of situations. A major challenge in distributed control systems is to cope with the distributed nature of events and the lack of central interventions in a dynamic environment.

Continuous advances in hardware, software and communication components increase the expectations on the performance of the robotic systems in terms of flexibility, reliability and responsiveness, by distributed control design that gives more autonomy to robot controllers and other field-level controllers with self-monitoring and maintenance properties. Many natural and manmade systems have hierarchically evolved in such a way that control is exercised in the most economical manner towards hierarchically autonomous self-organizing systems, as a major field of bio-inspired artificial intelligence. However, it is not easy to predict an emerging behavior resulting from a set of local interaction rules in a distributed control system. It is more difficult to identify the rules behind an observed global behavior, such as a component (cell)-based algorithm of self-assembly of arbitrary 3D shapes in analogy with an embryonic development process under gene regulatory networks.