Experiences with Middleware for a Networked Embedded Software Technology Open Experimental Platform

Venkita Subramonian and Christopher Gill
Department of Computer Science, Washington University, St. Louis, MO

Abstract

Networked Embedded Software Technology (NEST) [1] defines a class of Distributed Real-time Embedded (DRE) systems in which the number of computing nodes is large, typically between $10^2$ and $10^3$ nodes, and resources per node are highly constrained. With the advent of small, robust, and inexpensive Micro-Electro-Mechanical Sensors (MEMS) and actuators, computing nodes of this scale are being used increasingly in fields like advanced avionics and space systems. This has prompted the need for collocation and embedding of processing and control software within or in close proximity to MEMS components with strict memory and processing constraints.

NEST systems involve design forces from a number of key areas which must be resolved at once – e.g., distribution of control, heterogeneous processing, time synchronization, resource allocation, group membership and voting, fault tolerance, and system level configuration management. Middleware approaches, such as Common Object Request Broker (ORB) Architecture (CORBA) [2] ORBs can serve to encapsulate the inherent complexity of these design forces. However, middleware solutions for NEST appear highly dependent on the particular application. To address the specific design forces for each application, we therefore take a two-fold approach: we will (1) identify and apply patterns and pattern languages for NEST middleware, and (2) use advanced generative [3] programming techniques to re-factor the middleware frameworks that reify these patterns, for maximum flexibility.

Patterns we have identified so far as candidates for NEST middleware [4] include Lookup [5], Leasing [6], Lazy Acquisition [7], Blackboard [8], Broker [8], Distributed Observer [9], Distributed Scheduler [9], Replication Manager [10], Fault Detector [10], and Fault Notifier [10]. We will examine these patterns for applicability in the NEST environment, and generate both a pattern catalog and generative pattern languages for specific NEST application sub-domains. We also plan to mine patterns in other related technology areas, e.g., Jini [11], JavaSpaces [12], the Real-Time Specification for Java [13], and the OMG Smart Transducers RFP [14] and response [15].

References


