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

Venkita Subramonian and Chris Gill
Department of Computer Science
Washington University, St.Louis, MO
{venkita,cdgill}@cs.wustl.edu

Research supported by DARPA (NEST), Boeing OEP contract
What is NEST?

- Networked Embedded Software Technology
- Distributed Real-Time system with 100 to 100,000 networked nodes
- Resource constrained hardware components
- Requires fine-grain fusion of hardware and software components
- Applications in advanced avionics and space systems, weapon systems, wireless devices
NEST Services

- Predictable and dependable behavior despite local failures
- Real-time Coordination Services
  - Data exchange
  - Synchronization
  - Self-stabilizing protocols
  - Replication
  - Fault tolerance
- Automated synthesis of services
An Open Experimental Platform for NEST

Active Damage Interrogation

- Acoustic Waves (kHz Range)
- Structure with Embedded or Bonded Piezoelectric Transducers
- Actuator Excitation
- Information Processing System
- Sensor Measurements
Why Middleware for NEST?

- Service reuse across NEST applications
- Flexible framework
  - Can be customized to a particular NEST application/execution context
  - Can exist across various levels of scale
- Address NEST design forces through
  - Distribution of control
  - Resource management
  - Time synchronization protocols
  - Heterogeneous processing
  - Dynamic reconfiguration
  - Fault detection and recovery
Yet Another Middleware?

- NO!!!
  - No single solution will work across use-cases
  - Need an open framework within which we can compose only the NEST services needed
  - Requires fine-grained ACE-level primitives
- CORBA based middleware is well proven
- Leverage ideas from small footprint ORBs
  - e-ORB, UBI-core, etc.
- Use design patterns
  - Capture solutions to design forces in a context
  - Guide generative composition of primitives
NEST middleware composition

- ACE/TAO
  - Pattern rich middleware frameworks
  - Capture some inherent structure of the NEST domain

- Bottom-up approach
  - Re-factor ACE classes for finer granularity
  - Composition of features across multiple use cases

- Top-down approach
  - Subset TAO to meet NEST requirements
  - Coarser-grained and larger-scale, may be automated

- Hybrid approach
Towards a Fine-Grained Substrate

Decoupling concerns

- Reactor
- Acceptor
- Connector
- Event Handler
- Svc Handler

ACE_Event_Handler
ACE_Service_Object
ACE_Task_Base
ACE_Task
ACE_Svc_Handler

Peer stream
Conclusions and Future work

- Minimal footprint IIOP ORB framework
  - Full CORBA compliance both attainable and optional
- Development underway using and extending
  - NEST design forces (guide what is needed)
  - TAO strategies (capture key solutions)
  - ACE primitives (provide a flexible substrate)
- Will leverage advanced techniques for subsetting and extension
  - Generic/Aspect-Oriented/Generative Programming
  - Automated custom generation that leverages the evolution of the baseline