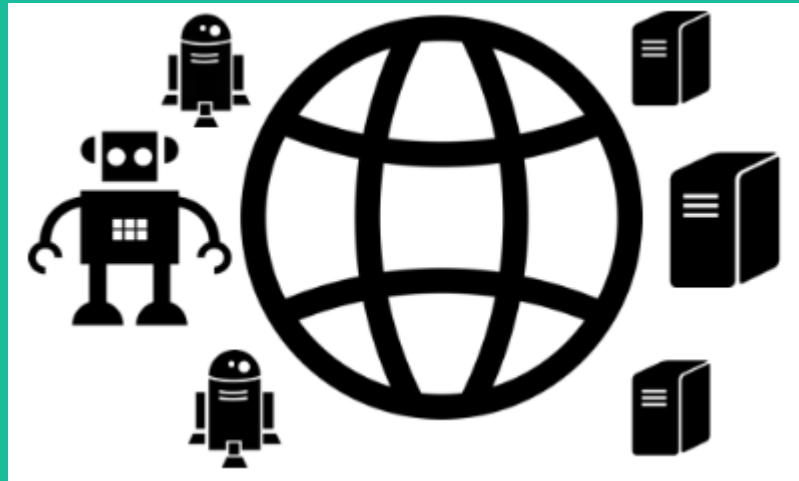


# Reference Middleware Architecture for Real Time and Embedded Systems: A Case for Networked Service Robots



Παππάς Γιώργος – Εφραίμ 9124

# Introduction:

- What is Middleware? (It can be described as "software glue".)
- Why is it important in networked service robots?
- Robot Middleware Requirements.
- Reference Robot Middleware Architecture.
- Future Challenges of Robot Middleware.

# Middleware Architecture in Networked Service Robots

- A networked service robot represents a contemporary real-time and embedded system.
- It is composed of a number of embedded processors, hardware devices, and communication buses making it a self-contained distributed system.
- Complex integration of these devices makes middleware architecture necessary.

# Robot Middleware requirements:

- Slow development progress of intelligent service robots till now.
- The low – cost / high - performance CPU's of today and the evolution of communications, allow us to use remote servers for complex calculations.
- A distributed system like this demands a sophisticated middleware to ease the management of the robot system.

# 1. Device abstraction and component models

- A robot is composed of a wide variety of hardware and software modules with distinct idiosyncrasies
- A robot system has to accommodate independently developed applications
- Those applications should be made unaware of low-level devices (they rely on)
- So the Middleware has to effectively encapsulate and parameterize a wide variety of devices inside the robot. (*device abstraction*)

•As robot is a part of an overall distributed system (robots, remote servers, home network appliances), the middleware has to support *component technology* as well as procedural and object – oriented communication:

- To hide heterogeneity in network protocols, OS's, implementation languages in distributed computing
- To support plug and play of software modules
- To “piece together” *device abstraction* layer and any upper - level software layers

## 2. Dynamic Reconfigurability

- A robot needs a new set of features every three weeks.
- The networked service robot is prone to network connectivity problems (remote servers take over heavy calculations)
- Robot middleware can support dynamic reconfigurability at three levels:
  - System level
  - Application level
  - Component level

### 3. Resource Frugality for Scalability

- Real – time and embedded systems can't enjoy all the benefits of current middleware technology due to low hardware capabilities and limited time window.
- Thus, we need to design a middleware specifically customized to a given application domain, like AUTOSAR
  - AUTOSAR has been widely adopted thanks to its compact and efficient runtime system and its static binding of components



## 4. Real – Time and QoS Capabilities

- A networked service robot provides services mostly with a strict deadline

Real – time vision and voice processing has a limited time window

- The robot middleware has to provide support for real-time guarantees and QoS.

# Reference Robot Middleware Architecture

.One example of robot middleware is the robot software communications architecture (RSCA).

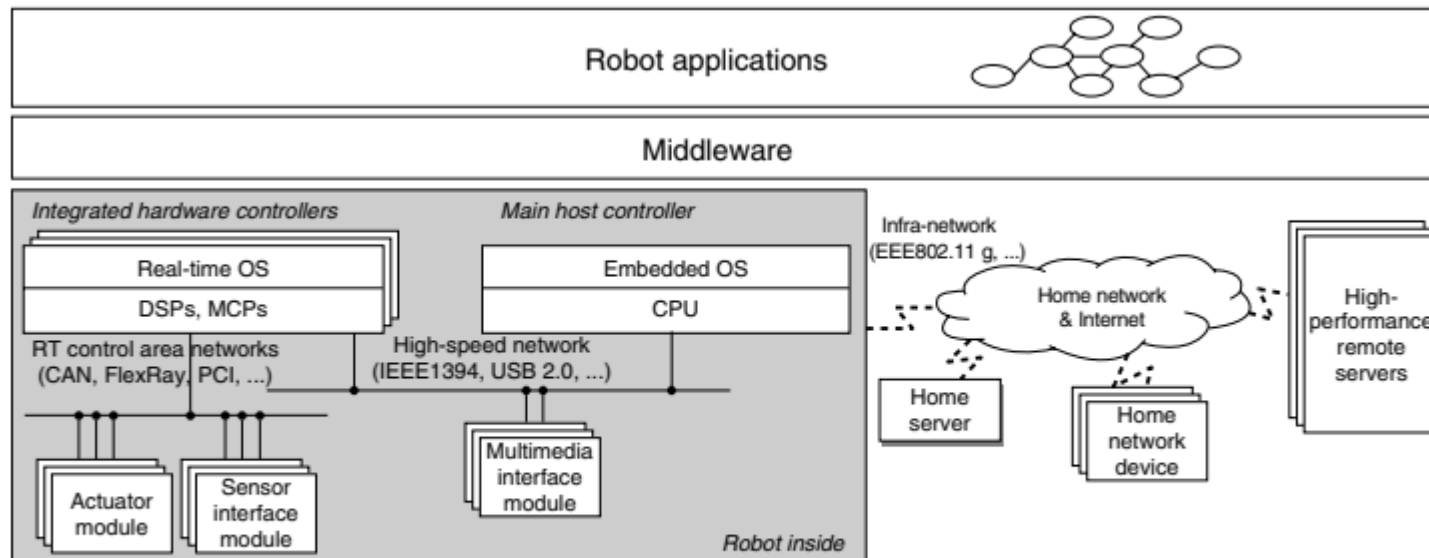


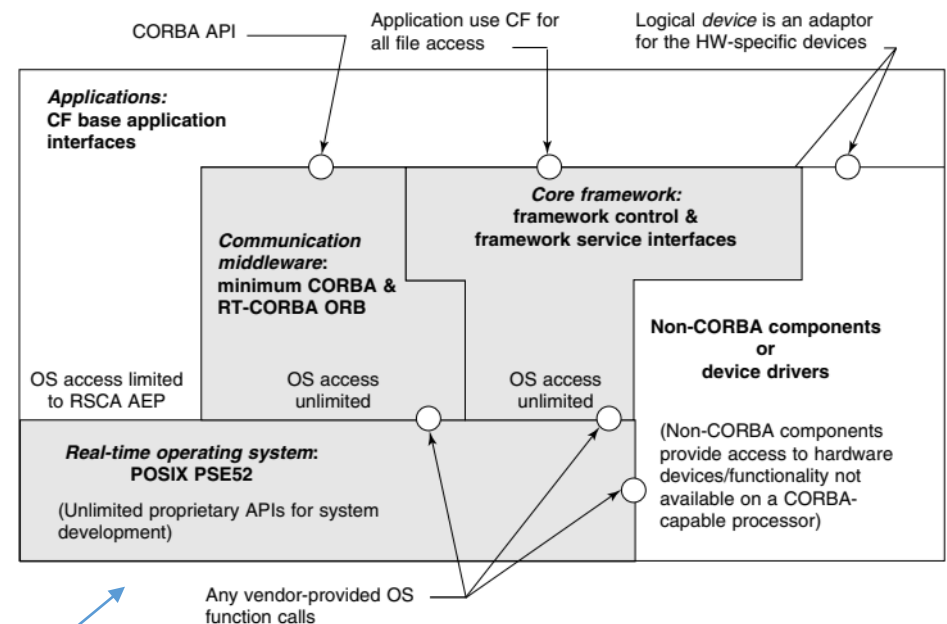
FIGURE 18.1 Reference structure of hardware and software of a networked service robot.

# Overall Structure of RSCA

The RSCA is specified in terms of a set of common interfaces for robot applications.

Standard OE interfaces (Defines the APIs developers use to deploy and control applications and exploit services from underlying platforms)

Standard application component interfaces (Defines interfaces that an application component should implement to exploit the component-based software model)



Overview of the operating environment of RSCA.

The RSCA's OE consists of a <sup>(1)</sup>real-time OS (RTOS), a <sup>(2)</sup>communication middleware and a <sup>(3)</sup>deployment middleware called Core Framework (CF)

(1) The RTOS provides a basic abstraction layer that makes robot applications both portable and reusable on diverse hardware platforms

(2) The communication middleware is an essential layer that makes it possible to construct distributed and component-based software

Specifically the RT-CORBA compliant middleware provides:

1. A standard way of message communication
2. A standard way of using various services
3. Real – Time capabilities

(3) The deployment middleware layer provides a dynamic deployment mechanism by which robot applications can be loaded, reconfigured, and run.

# RSCA Core Framework

In RSCA a robot system is a domain that distinguishes each robot system uniquely.

In a domain there exist:

- Multiple processing nodes -> They serve as units of hardware reconfigurability.
- Multiple applications -> They serve as units of software reconfigurability.

It is achieved by attaching or detaching a node to or from the domain.

It is achieved by creating an instance of an application in a domain or removing the instance from it.

# Core Framework Interfaces

## CF interfaces consist of three groups of APIs:

1. Base application interfaces.

Interfaces that the deployment middleware uses to control each of the components comprising an application.

2. CF control interfaces.

Interfaces that include the functionalities of starting and stopping the resource, configuring the resource, and connecting a port of the resource to a port of another resource.

3. Service interfaces.

Common interfaces that are used by both deployment middleware and applications.

Among these interfaces, ResourceAllocator and EventChannelManager are interfaces defined for QoS and distributed event services, respectively.

- ResourceAllocator with domain profiles allow applications to achieve desired QoS guarantees by specifying their requirements in the domain profiles.
- EventChannelManager with domain profiles allow applications to make connections via CORBA event channels by simply specifying their connections in domain profiles.

Domain profiles are a set of XML descriptors, describing configurations And properties of hardware and software in a domain.

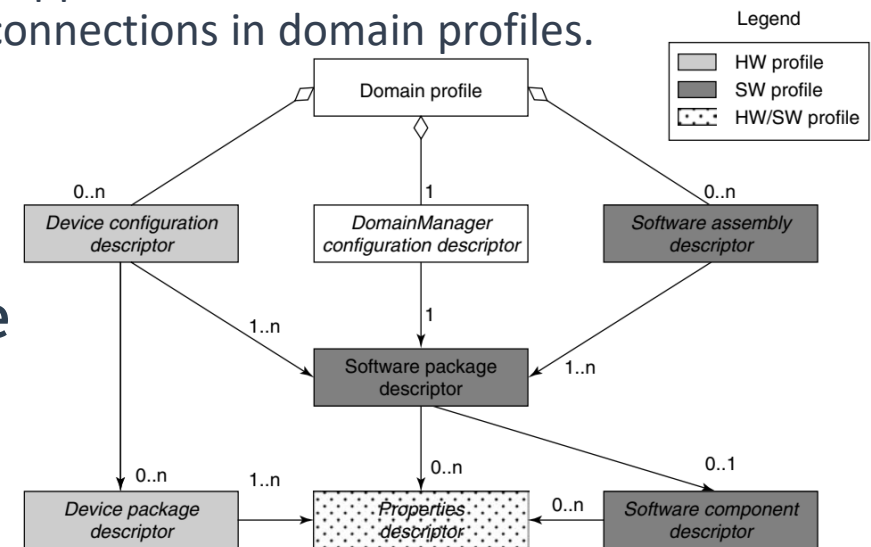



FIGURE 18.6 Relationships among domain profiles.

# Future Challenges of Robot Middleware

To develop commercially viable middleware for RTES, researchers must address technical challenges that arise due to future trends in embedded systems.

The 2 main trends are:

- An embedded system will become a heterogeneous distributed system as high performance computations are required
  - Applications that used to run on high-end computers will be deployed on embedded systems for enhanced user interfaces
- 

---

Future technical challenges for real-time embedded middleware:

- Performance optimization via new metrics
  - Streaming support for high-volume data
  - Provision of domain-specific abstractions



# Performance Optimization via New Metrics

## Cost:

- Historically, the most critical hurdle that robot industry has been facing is the high cost of robot, compared to the level of its intelligence.
- The internet-implemented robot intelligence is a system redesign and an optimization that provides better value for money ratio.

## Energy:

- Embedded supercomputers are mobile systems working on a battery.
- Reduction in power consumption of embedded systems is critical design aspect
- Thus, embedded system developers should judge their systems via performance-power-cost product

# Streaming Support for High-Volume Data

Most applications running on RTES involve streaming of extremely large volume of data.

Application examples:

- Real-time vision
- Speech recognition
- Multimedia presentation

Future real time and embedded middleware should support a streaming data model and a data synchronization model for stream dataflows

# Domain-Specific Abstractions

Lack of domain-specific abstractions makes the life of robot developers hard, as their programming abstractions differ from those of existing robot software frameworks.

Robot software frameworks:

- specialize in modeling sensor and actuator devices and executable devices (DSPs, MPUs).
- provide abstraction models that capture robot behavior architectures

To help the adoption of robot middleware it is desirable to integrate existing robot software frameworks into robot middleware.

# Conclusions

-Slow adoption of middleware into commercial real-time and embedded systems due to resource limitation, reliability and cost requirements

-RSCA, is a reference middleware architecture for RTES, providing features like:

- Device abstraction model and Component model
- Dynamic reconfigurability of a robot system
- Resource frugality
- Real – Time and QoS capabilities

-RSCA is not perfect. Future trends will bring new technical challenges and new requirements will arise.