Using CCM to Develop Resource Status Service (RSS)

Rich Shapiro
BBN Technologies,
Cambridge, MA
&

Balachandran Natarajan
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee

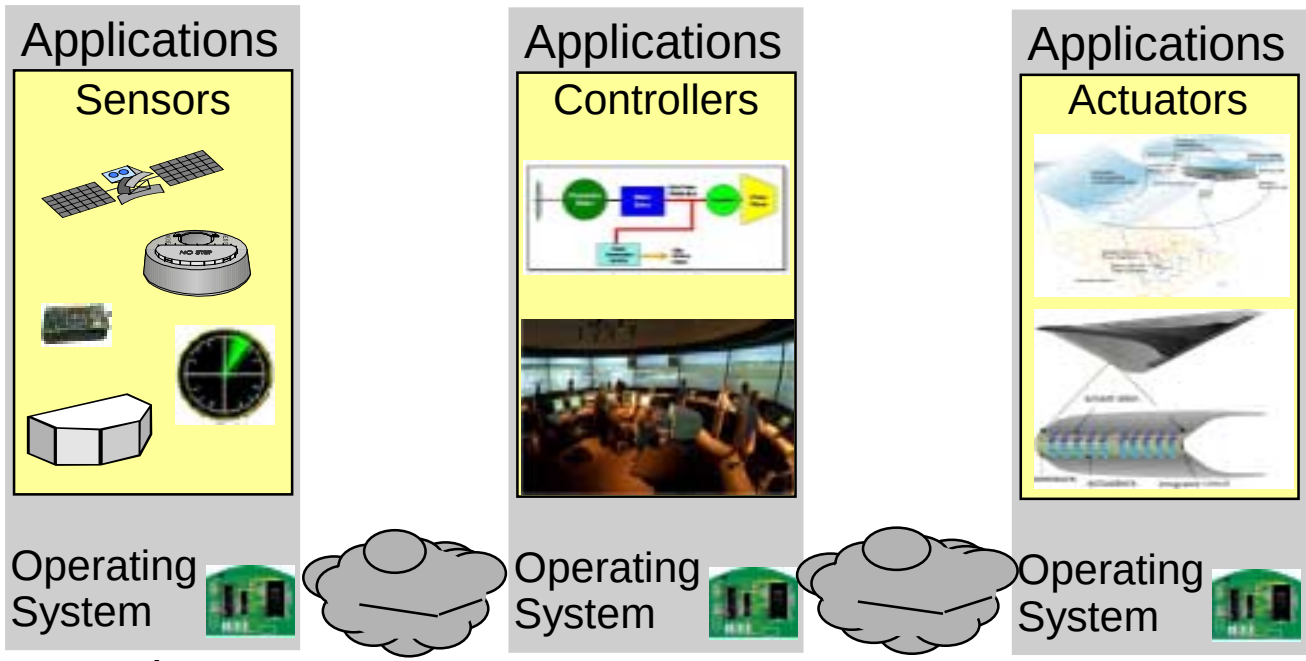
Sponsored by Defense Advanced Research projects Agency (DARPA) under contract number NBCHC030119

Motivation

We increasingly rely on distributed real-time & embedded (DRE) systems

Characteristics of DRE Systems

- **Network-centric**
- **Stringent quality of service (QoS) demands**
- **Resource constrained**
- **Mission-critical control**



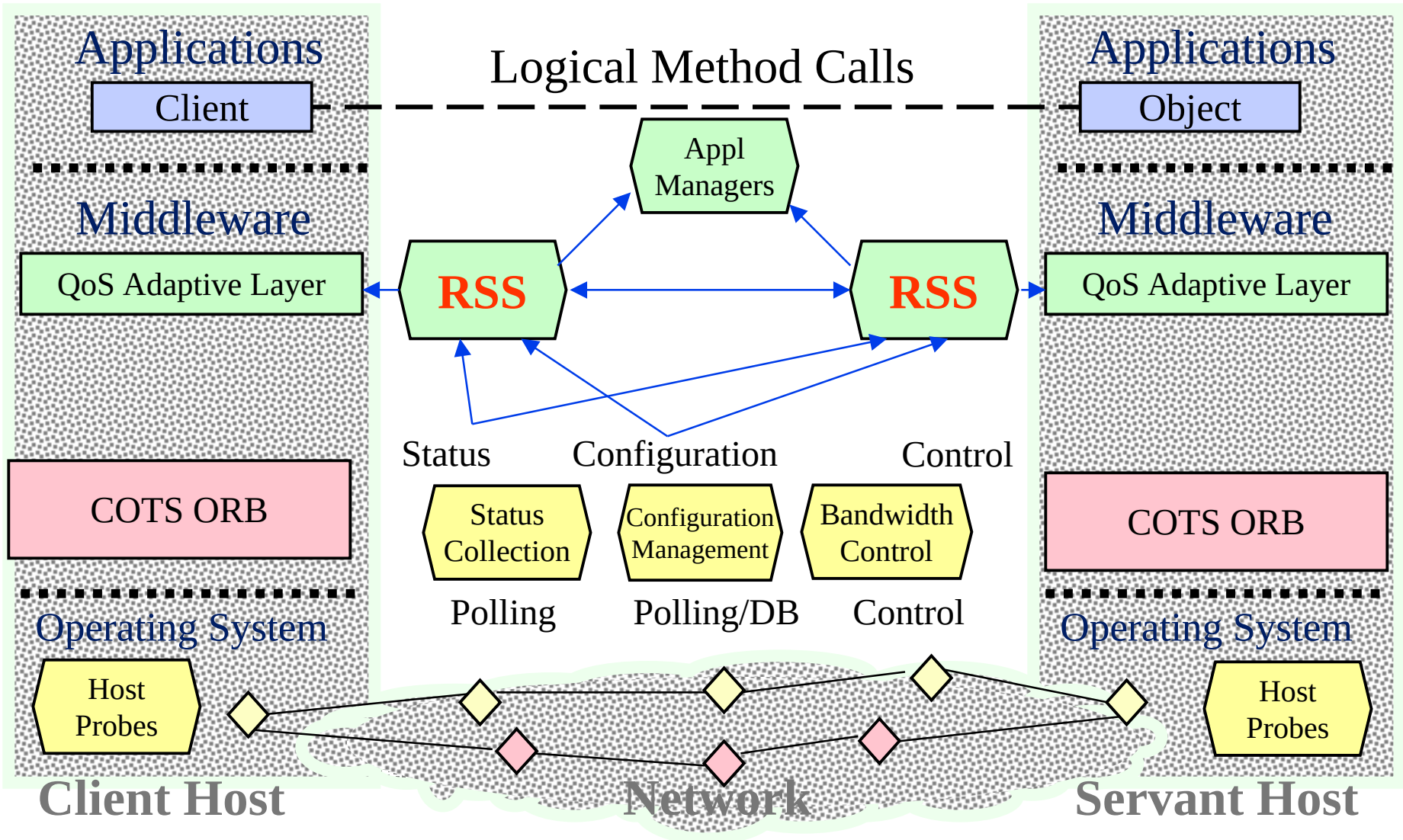
New Challenges

- **Adaptive DRE Systems**
- **Reflective systems**
- **Efficient resource allocation and utilization**
- **Ubiquitous and time-critical systems sharing resources.**

Resource Status Service

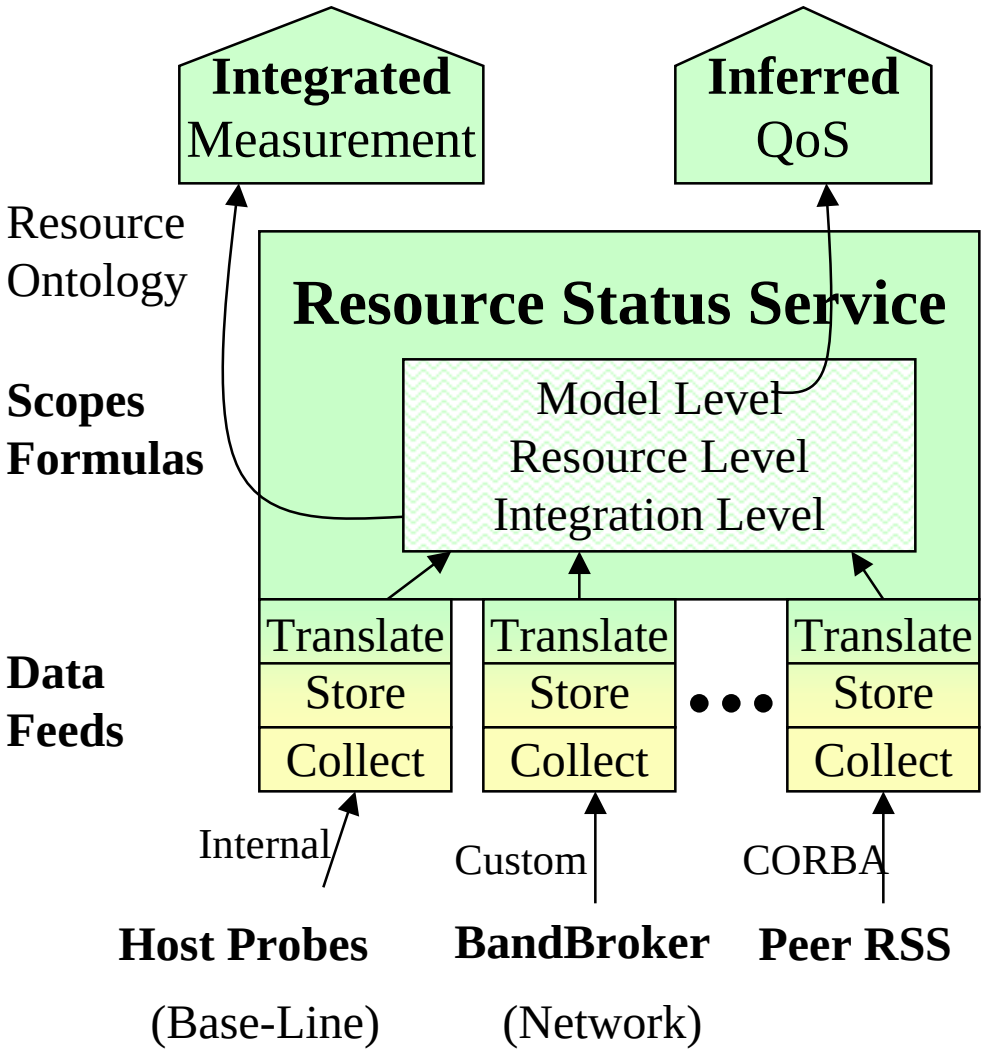
- A distributed service handling distributed QoS data
- Dynamic adaptation to changes in resource availability in a DRE system requires
  - timely, accurate and relatively high-level integrated data.
  - has to be synthesized from a set of distributed simpler data
- Federated service with each node or object in the federation handling
  - Incarnates a particular data model.
  - Raw sensor data and handling requests for integrated data

Resource Status Service



STATUS OF THE WORKING IMPLEMENTATION

Properties



Integrates external data about remote objects to infer expected resources

Isolation: Clients work with a high level description of available resources

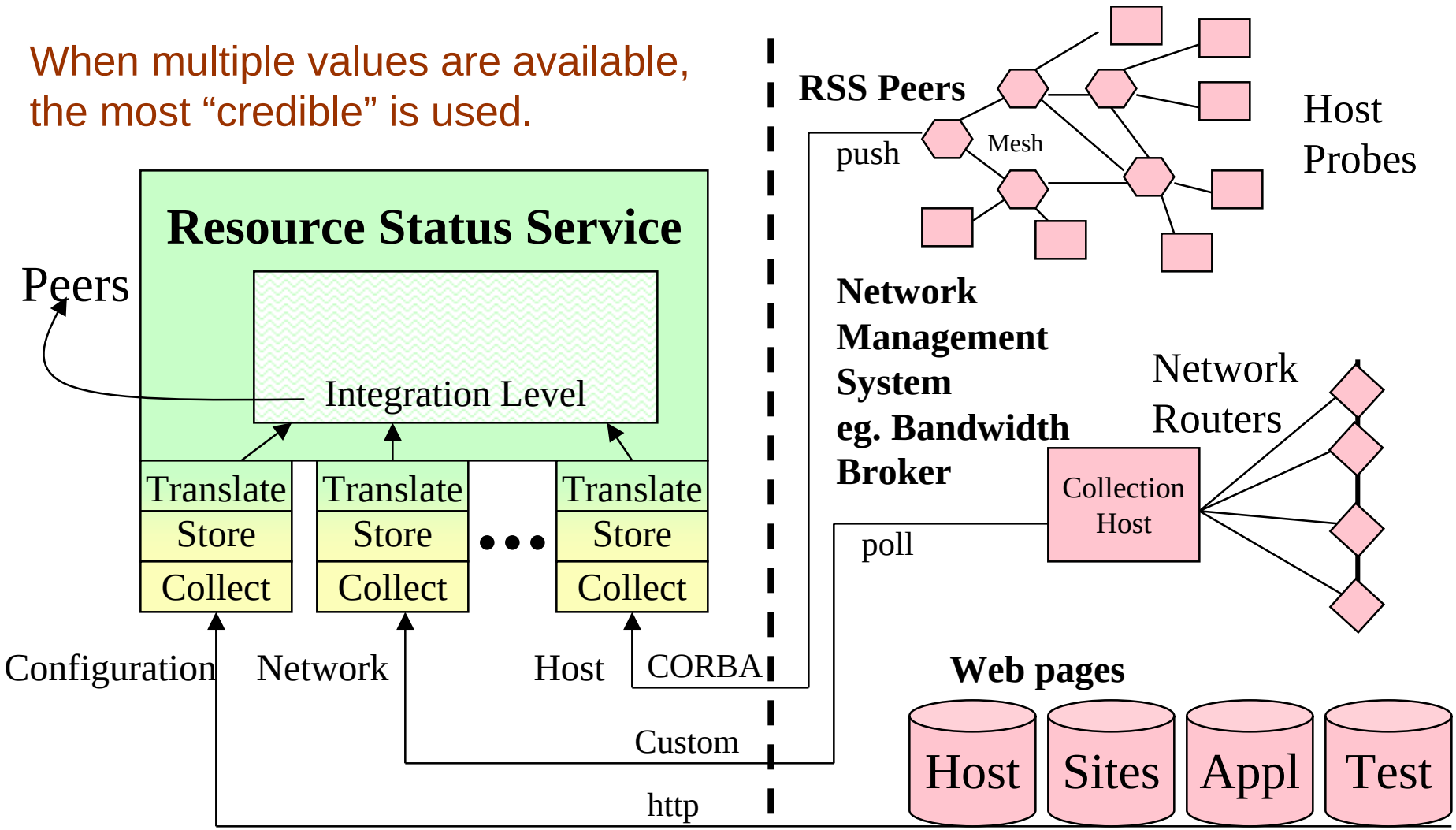
Integration: Conflicting measurements will be resolved to always give the best guess.

Translation: different standards for Resources MIBs will be translated into a Resource Ontology

Collection: interfacing details will be handled by Data Feed

RSS with Data Feeds

When multiple values are available, the most “credible” is used.



Data Feeds interface to a specific source of system data.

are Organized into a QoS MIB (Key : Value Pairs)

Host_10.23.12.1_**CPU**_LoadAverage_Expected
Host _10.23.12.1_**Process**_2232_**Socket**_5223
Host _10.23.12.1_**Process**_2232_**Object**_Key_tao/foo/2
IPFLOW_ 10.23.12.1_ 10.23.12.1_Capacity_Max

- **Keys** form a hierarchy of names by convention
  - Root is an entity
  - Leaves are QoS Properties
  - Hierarchy based on Existing Standard MIBs,
    - WBEM/CIM, SNMP, QoSMIB
- **Data Values** have multiple attributes
  - Value (dynamic type, such a boolean, int, double, string…)
  - Credibility (belief on how good the measurement is)
  - Units, Timestamp, collector name, etc)

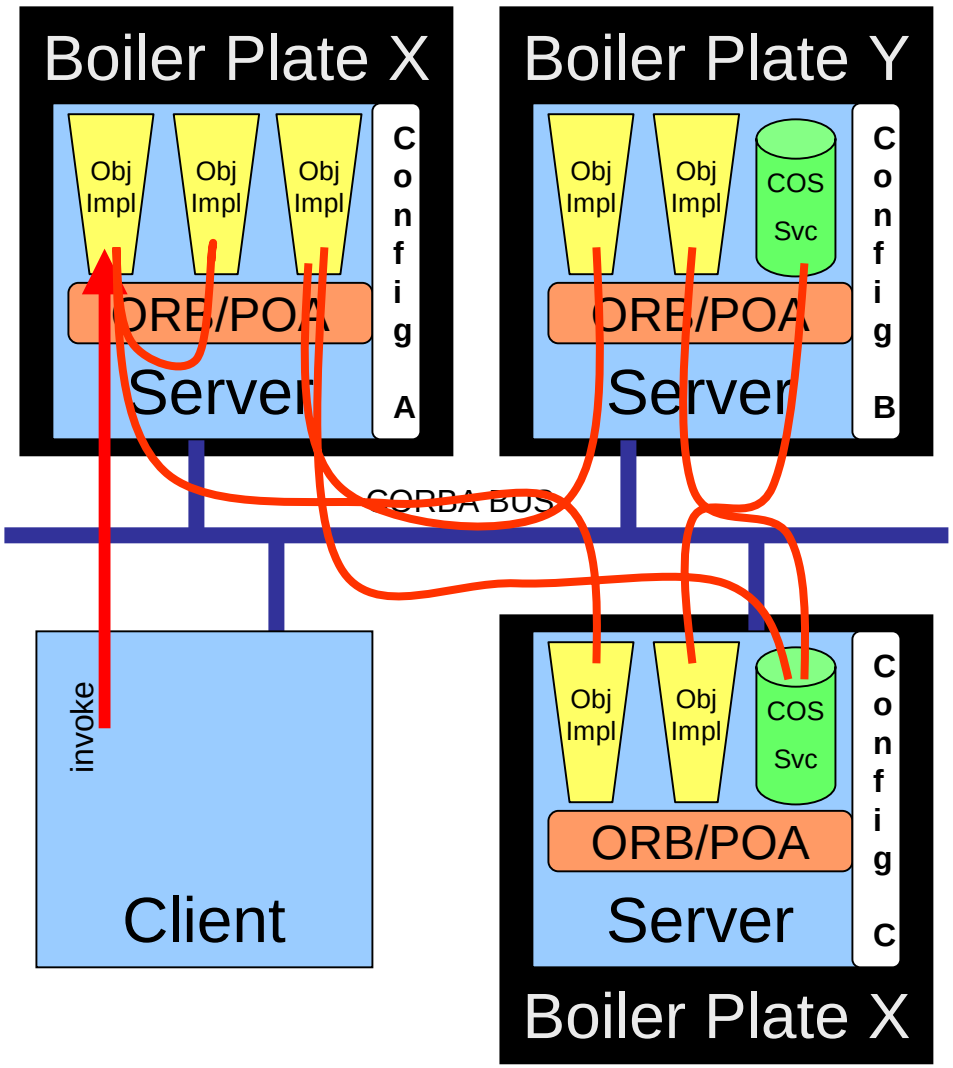
Publish and Subscribe Network

- **Formulas** Subscribe to values relative to the model scopes for which they are attached
- **Formula Setup time** uses model scopes to find the sub-formulas.
  - The expensive setup happens only once.
- **Value Calculations** are done continuously as raw data changes
- **Both** forward chaining and backward chaining modes are supported,
  - Pull vs Push
- Formulas are extremely **Reusable**
  - Attach them to a scope and they will calculate a value

Instances Propagate Raw Data to Peers

- Host to Pool
- Application to Application String

Limitations of CORBA 2.x Specification



- Requirements of non-trivial DOC applications:
  - Collaboration of multiple objects & services
  - Deployment on diverse platforms
- Limitations – Lack of standards for
  - Server configuration
  - Object/service configuration
- Consequence: tight couplings at various layers
  - Brittle, non-scalable implementation
  - Hard to adapt & maintain
  - Interfaces are forced by deployment decisions

Limitations of Object Model on RSS

- Hierarchical deployment of RSS is not trivial
  - For example, RSS at every level needs to be told about its environment
  - Attributes and operations added that has less relevance to the functional aspects of RSS
  - RSS at every level needs to discover and connect to its peers.
- Redeployment of RSS is non-trivial and could involve stopping and restarting applications.
- No standard way for clients to connect and subscribe to QoS data.

Deploying RSS hierarchicaly

- Problem: Hierarchical deployment of RSS not easy.
- Context: Deploying RSS along with applications hierarchically for a group of applications at every level in the hierarchy is hard
- Solution: Having the configuration data as metadata RSS to deployed and configured with ease. CCM provides basic mechanisms and tools to help with the same.

Subscribing to QoS Data

- Problem: No standard way for clients to subscribe to RSS data
- Context: Every RSS client needs to add code for subscribing to QoS data from RSS
- Solution: Use *`eventtypes`* in IDL3 to subscribe for QoS data. Use event filtering capabilities of the container.