
Customizable Middleware for Dynamic Distributed Environments

Nalini Venkatasubramanian

Distributed Systems Middleware Group

Department of Information and Computer Science

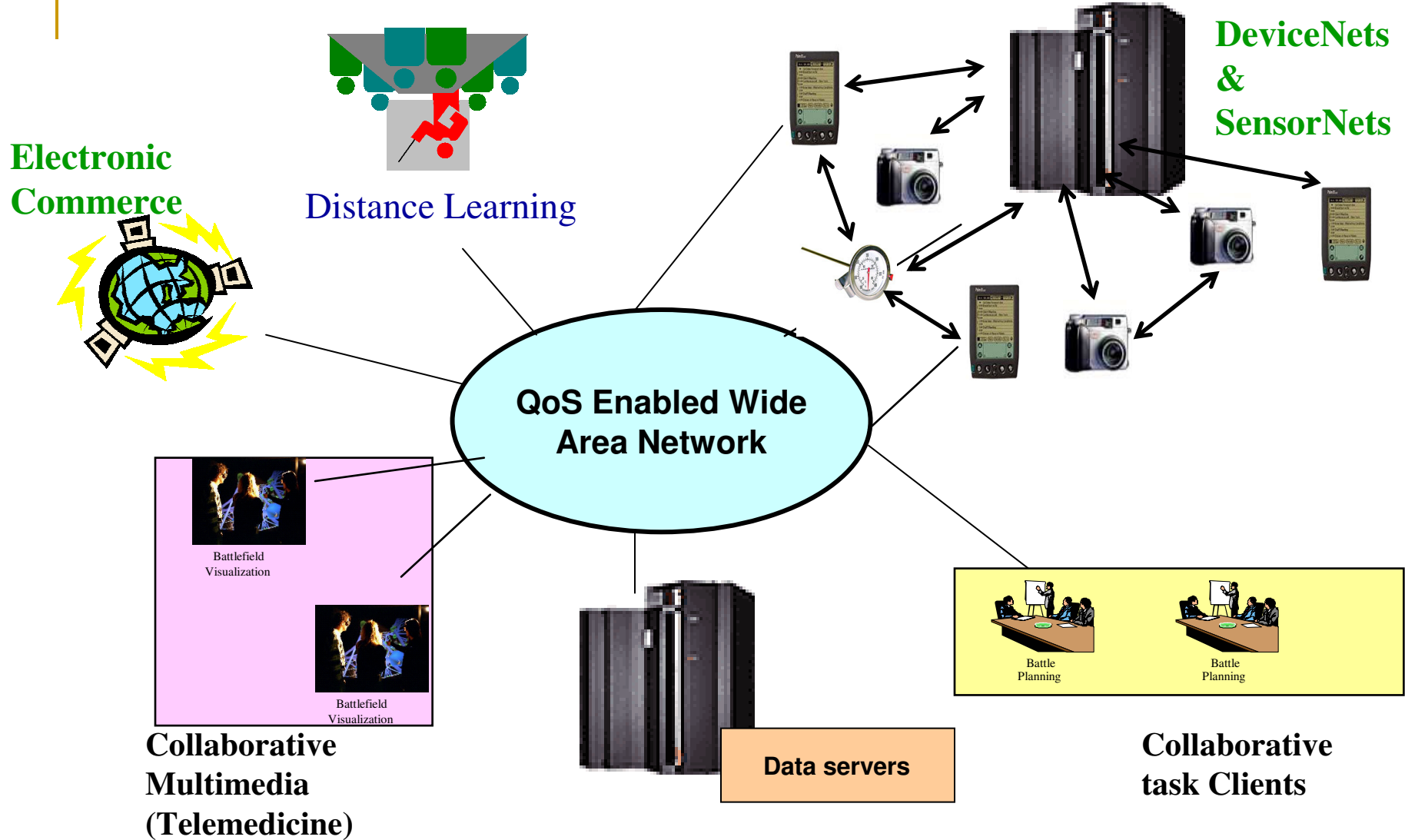
University of California, Irvine

UNIVERSITY OF CALIFORNIA, IRVINE



Distributed Systems Middleware Group

QoS Aware Information Infrastructure



Requirements - Availability, Reliability, Quality-of-Service, Cost-effectiveness, Security

Global Information Infrastructure

- Proliferation of devices
 - System support for multitude of smart devices that
 - attach and detach from a distribution infrastructure
 - produce large volume of information at a high rate
 - limited by communication and power constraints
 - Require a customizable global networking backbone.
 - Explore effective middleware infrastructures which can be used to support efficient QoS-driven resource provisioning algorithms in a highly dynamic environment
 - Middleware services have complex interactions
 - **risks include non-termination, information loss, inconsistencies and incorrect execution semantics**
-

Why Reflective Middleware?

- Wireless communication, mobile computing and real-time applications demand
 - **High adaptability**
 - **dynamic customization of systems, services, communication protocols**
 - **Safe flexibility**
 - **constrain composition of services and protocols in order to prevent functional interference that could lead to an inconsistent state of the system**
 - **required to protect the system from security threats and failure**
 - **Cost-effective QoS guarantees**
 - In achieving these goals, one must be careful to maintain consistency and correctness – need a semantic model (TLAM)
-

Composable Reflective Middleware

- In providing such an environment, interactions between various components as well as system-level correctness must be maintained
 - Incorporate effective mechanisms into a composable middleware framework to ensure safety and QoS enforcement in distributed and mobile environments.
 - Customizable, Composable Middleware Frameworks
 - Provide for dynamic network and system customization, dynamic invocation/revocation/installation of services
 - To adapt to the above dynamic changes in modern applications and manage distributed components
-

Reflection

- Provides a *plug-and-play* environment for enabling run-time modification of policies
 - An efficient technique to build composable middleware
 - Features
 - Separation of concerns
 - Introspection
 - Flexibility, Adaptability
 - Composition
 - Implies concurrent execution of multiple resource management policies
-

Composability Problems

- Access control and context awareness
 - Co-ordination protocols built into DS core service
 - Secure mobility
 - Dynamic delegation and revocation (keys, access rights, certificates) with mobile agents and hosts
 - Failures and mobility
 - Mobility + Security + QoS
 - Need approximate notions of correctness!
-

Composability Problem 1

- Customizable communication and mobility
- Snapshot + mobility



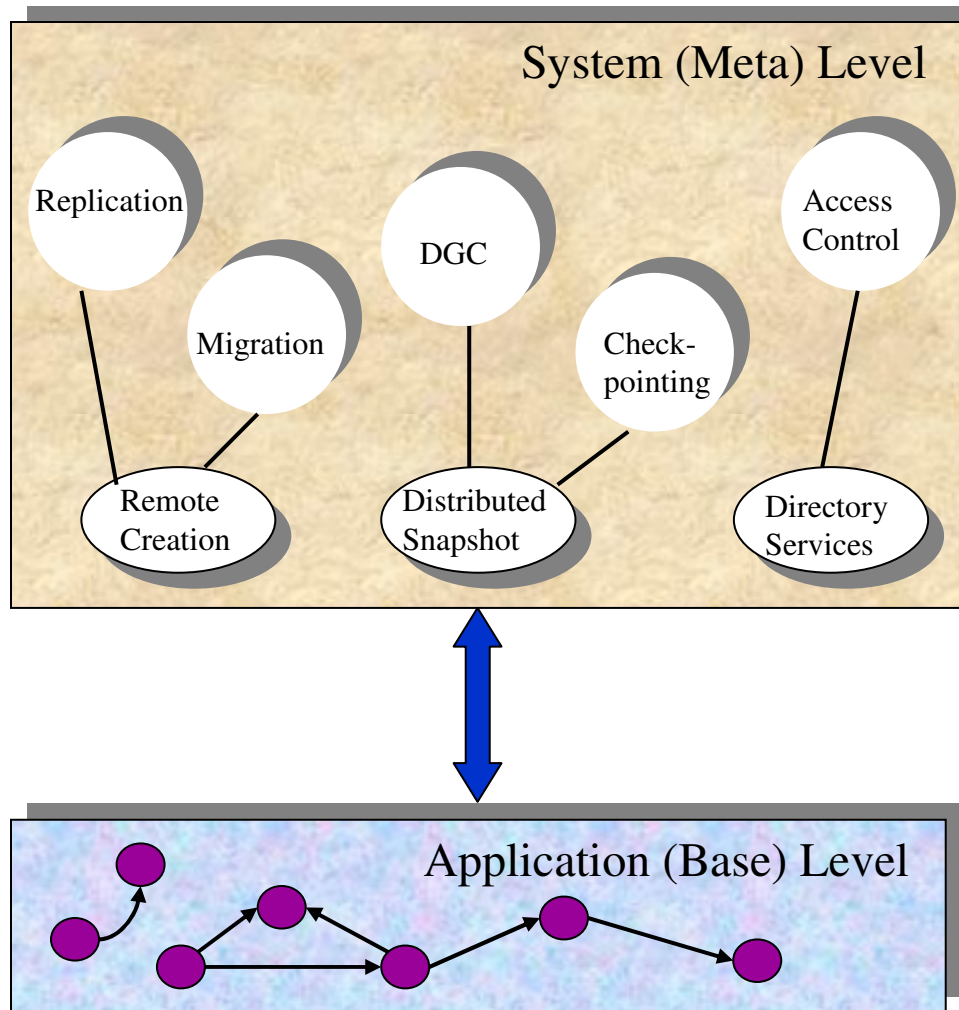
Composability Problem 2

- Security and Mobility
 - ❑ Support for disconnected entities
 - ❑ Efficiency
 - ❑ Restriction
 - ❑ Cascading
 - ❑ Concurrency (Mobility during the delegation / revocation process)
-

A Structured Approach to Reflective Middleware

- Two Level MetaArchitecture
 - Separate enforcement of system level requirements from application level activities to permit customizability
 - Layered Specifications
 - End-to-end service specification
 - System-level architecture specification
 - Local behavior specification
 - Isolate complex interactions in well understood *core services* for managing composition
-

Core Services

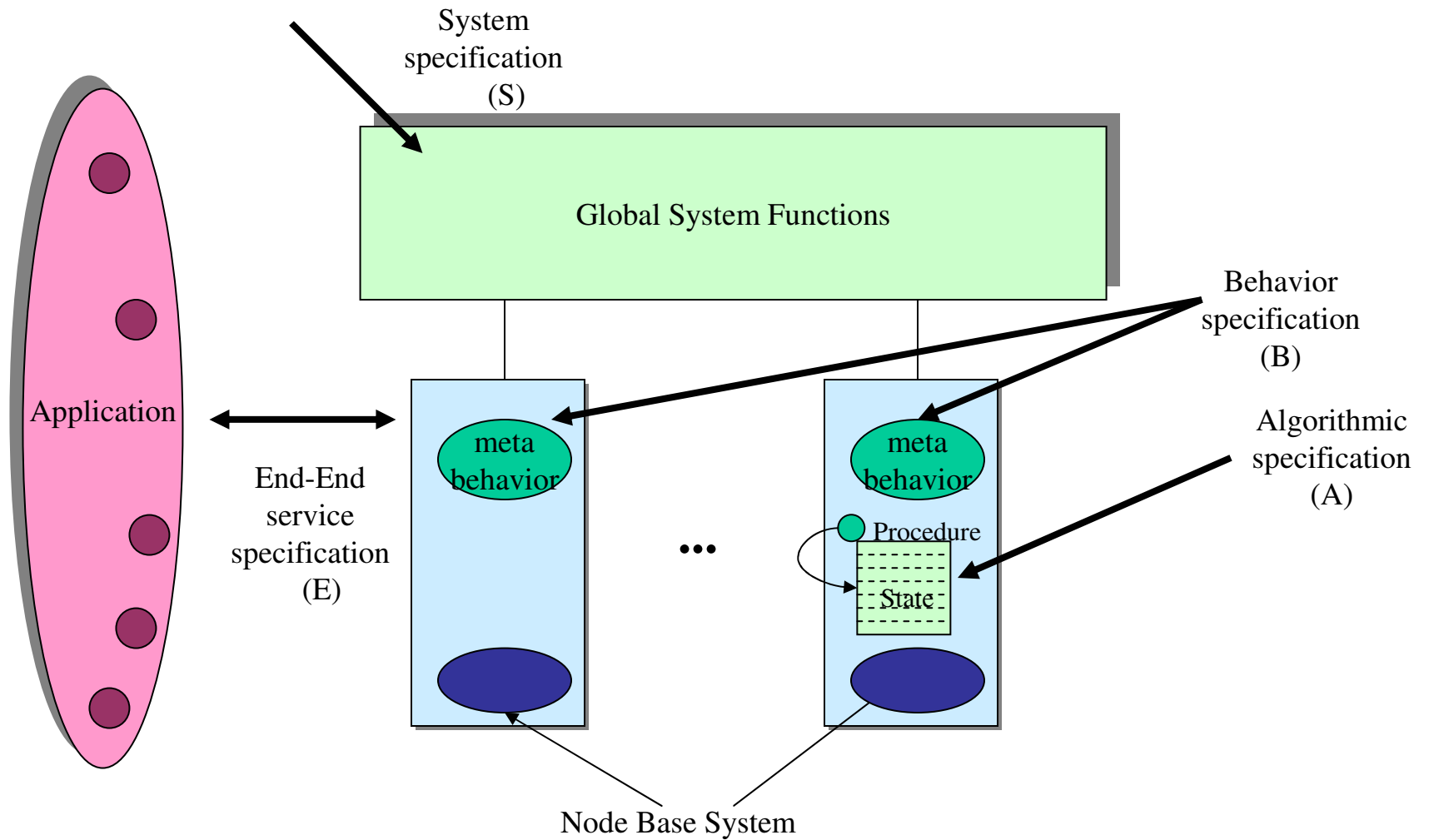


Core services allow us to isolate complex interactions
-- useful for managing composition of services

Two Level Actor Machine (TLAM)

- A semantic framework for specifying and reasoning about components of ODS (middleware services)
 - Features
 - based on the actor computation model for OODS
 - base-level actors model application functionality
 - meta-level actors model middleware services
 - use of core services to isolate interactions
 - specification viewpoints
-

Specification Viewpoints



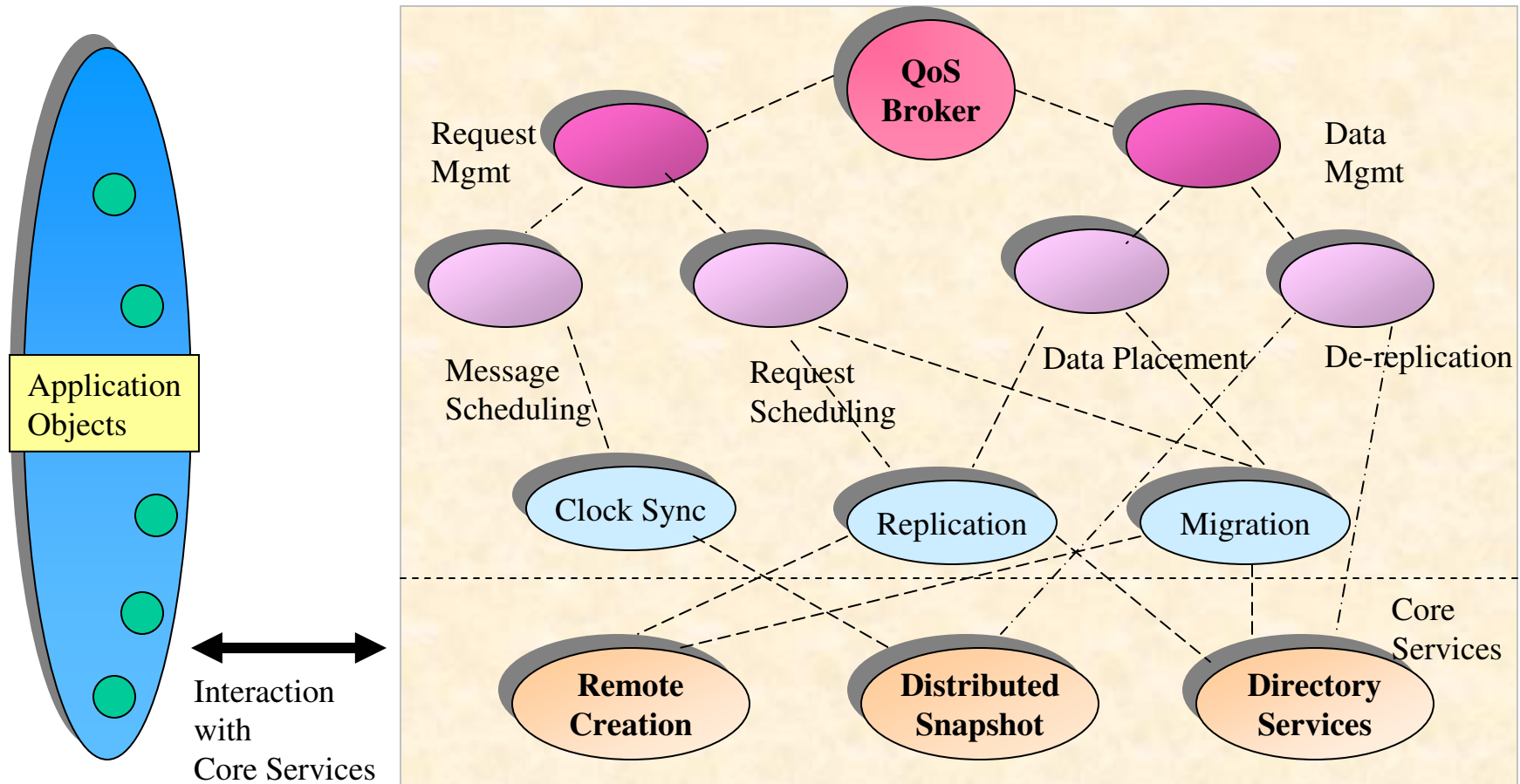
Relating Specification Viewpoints

- $(S \Rightarrow E)$ system spec implies end-to-end service spec
 - $(B \Rightarrow S \text{ if } I \text{ and } NI)$ behavior spec implies system spec if
 - (I) initial conditions satisfied
 - (NI) non-interference conditions satisfied
 - $(A \Rightarrow B)$ algorithm spec implies behavior spec
 -
-

QoS Broker

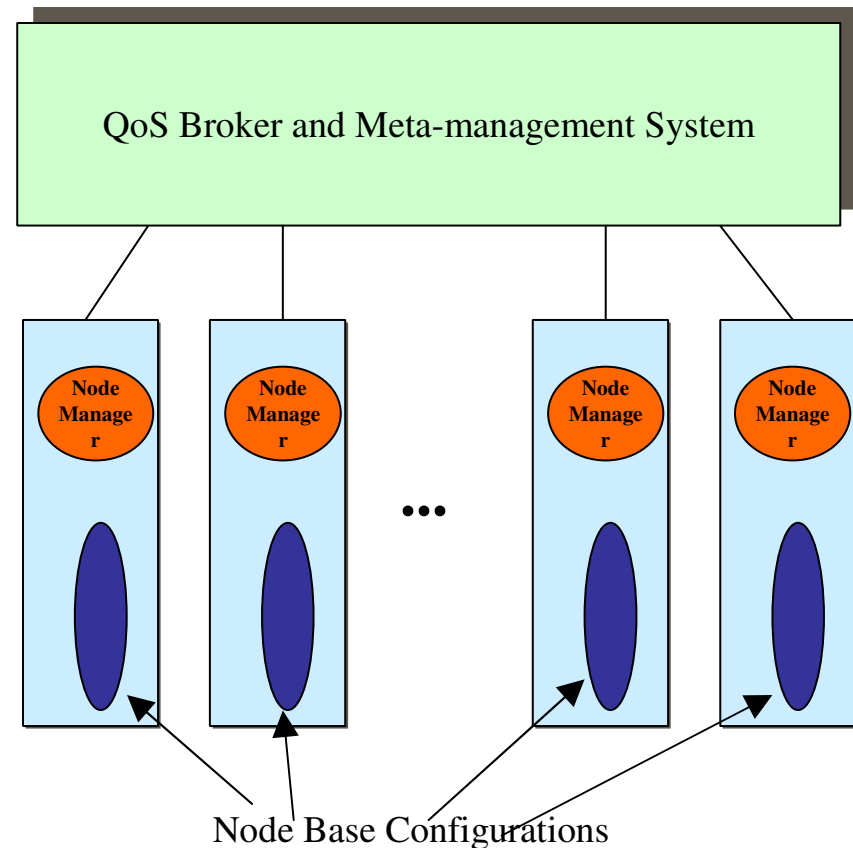
- Composability is essential in ensuring cost-effective QoS in distributed multimedia systems
 - Safe composability of resource management services
 - QoS Brokers coordinate multiple activities in such systems
 - interactions between multiple QoS sessions
 - interactions with multiple system services
 - Functions of a QoS broker:
 - Deal adaptively with incoming requests
 - Re(configure) data to service requests
 - Must maintain resource allocation invariants
-

The CompOSE/Q Framework

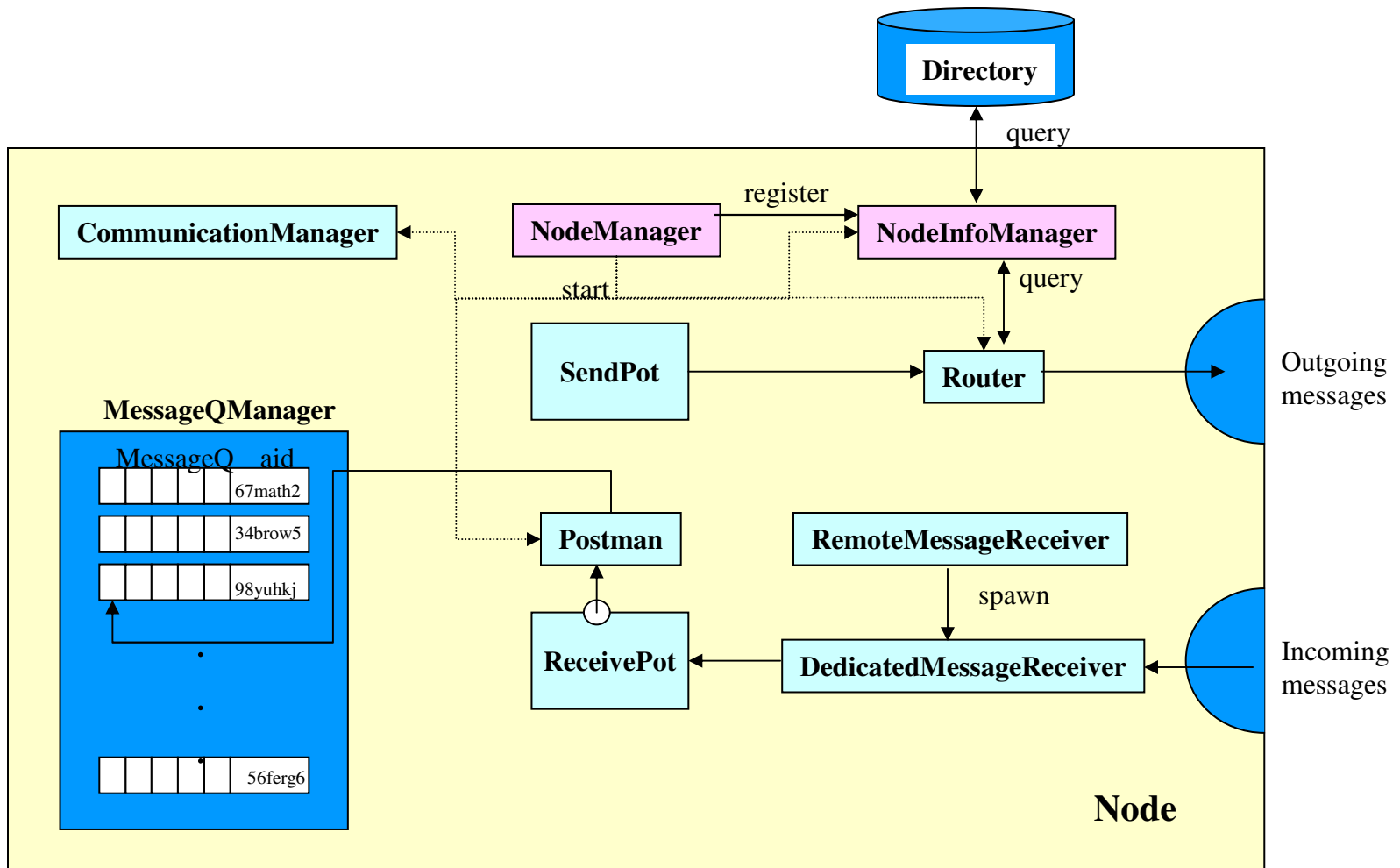


The ComposeIQ Framework

- **Physical Configuration:**
 - Nodes distributed across a network
- **Middleware Configuration:**
 - Distributed Component
 - Instance of runtime kernel on each node
- **Programming Environment:**
 - Based on concurrent objects

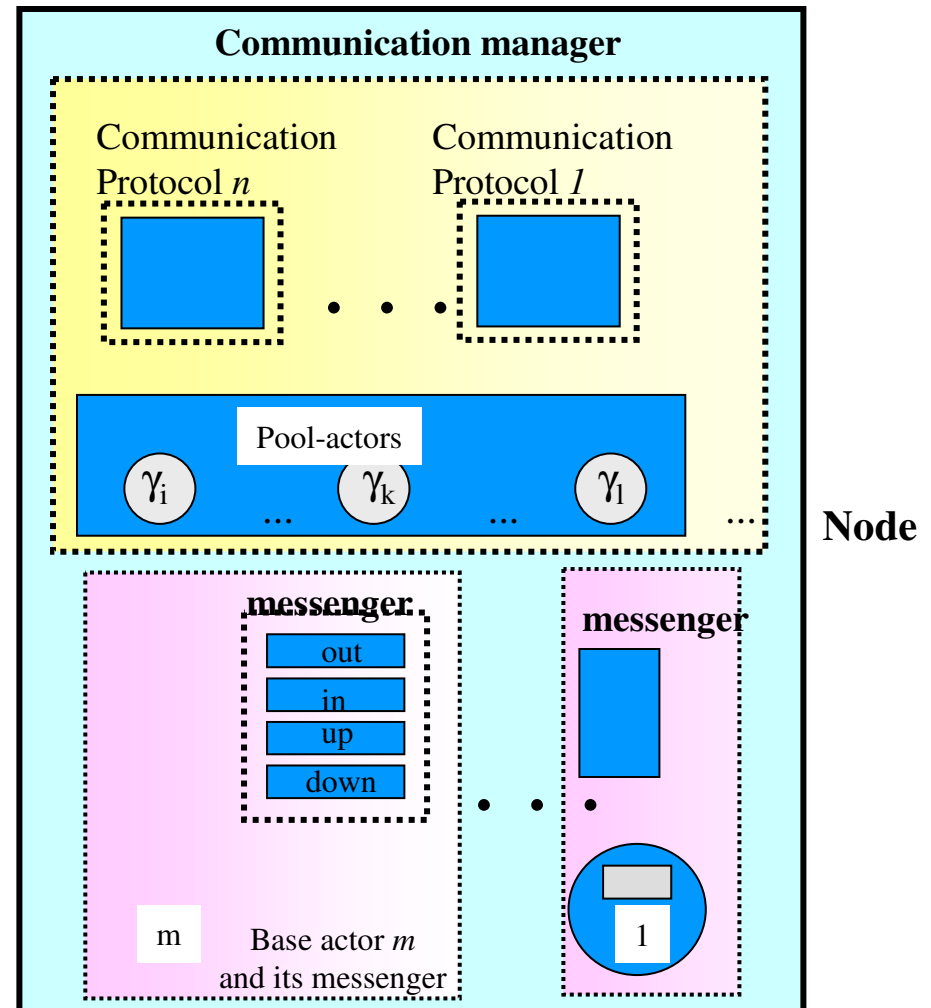


Compose1Q Node Runtime



Customizable Communication

- Distinguishes and handles different types of messages and communication protocols
- Integrates QoS parameters into resource management and message handling processes.
 - High-level communication services through composition of basic protocols
 - Dynamic installation of protocols
 - 4 levels of message customization
 - Efficient implementation

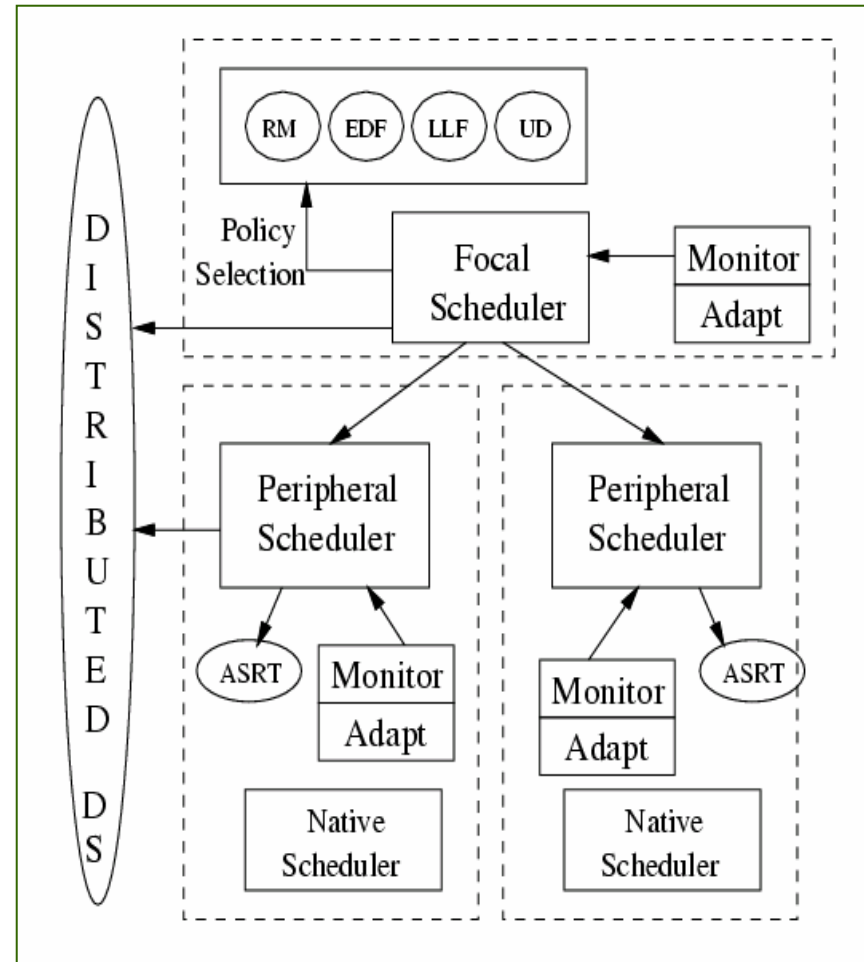


Customizable Security

- Capability-based access control architecture
 - Customizable Security Mechanisms
 - Adapt security policies to fit user needs under changing system and network conditions
 - Resource constraints, network bandwidth
 - Tradeoffs for Secure QoS
 - Domain-based Security
 - Inter-domain security based on various security levels.
 - Protocols to define object mobility
 - Customizable Authentication
-

Customizable Scheduling

- **Conjunctive Priority Assignments**
 - Assign priorities to both tasks and messages
- **Focal Scheduler**
 - Coarse grained scheduling
- **Peripheral Scheduler**
 - Fine grained scheduling



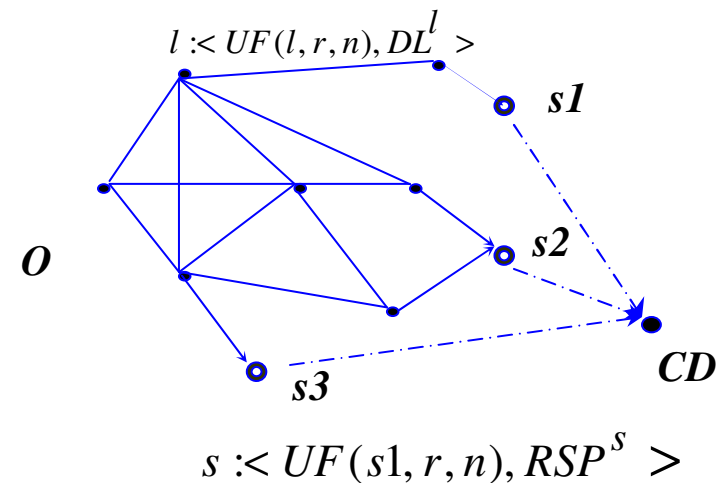
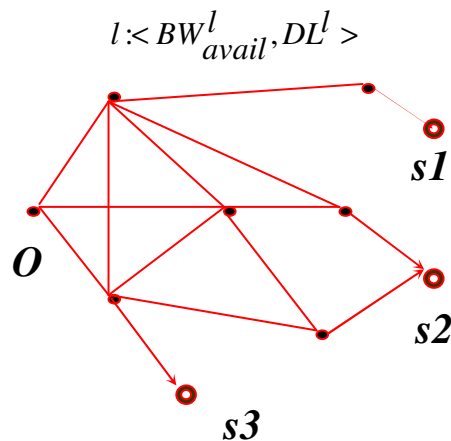
Dealing with Mobility

- Degree of network awareness that middleware and applications must have to deal with network conditions
 - Resource provisioning algorithms utilize current system resource availability information to ensure that applications meet their QoS requirements

 - Additional Challenges
 - In highly dynamic (e.g. mobile) environments, system conditions are constantly changing
-

QoS-provisioning in Mobile Environments

- Directory Service as a core service For QoS Based Resource Management in Mobile Environments
 - State information enables decision making for resource provisioning - e.g. Routing, Scheduling and Placement
 - Maintaining accurate and current system information is important
- CPSS (Combined Path and Server Scheduling)



Dynamic Service Brokering for Mobile Environments

■ Goal

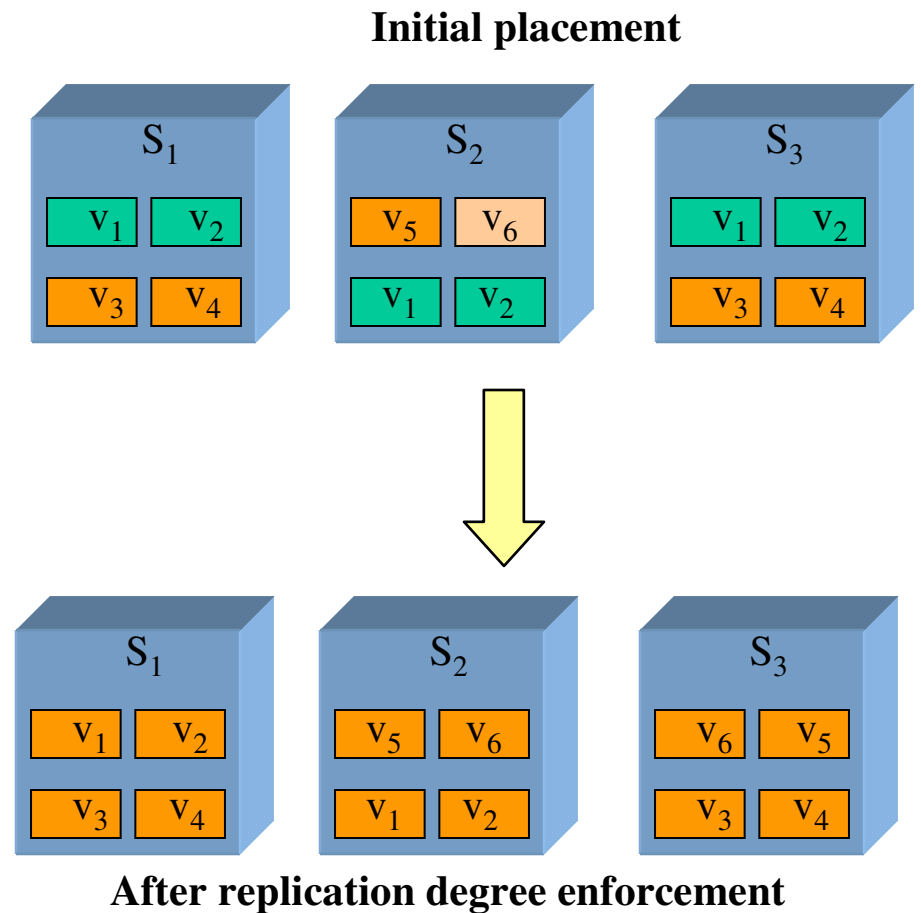
- ❑ To provide information good enough for resource provisioning tasks such as admission control, load balancing etc.
 - Need an information collection mechanism that is :
 - ❑ is aware of *multiple* levels of imprecision in data
 - ❑ is aware of quality requirements of applications
 - ❑ makes *optimum* use of the system (network and server) resources

■ Collected Parameters

- ❑ Network link status, Data server capacity (Remote disk bandwidth, Processor capacity)
-

Data Placement in Mobile Environments

- Design replication and intelligent data placement mechanisms that
 - Ensure effective resource management
 - Ensure QoS for admitted clients
- Design caching mechanisms
 - Partition data/processing between mobile clients and infrastructure
 - Allows disconnected operation
 - Efficient power management



Directory Services for Context Awareness

- The Directory Core Service facilitates agent-centric context awareness
 - Inherent support for attribute-driven searches
 - Resource Discovery protocols
 - Composability issues
 - Need formal notion of context correctness
 - How is this correctness preserved with mobility
 - Disconnected operations
-

Limitations of the TLAM

- In order to make initial progress in reasoning about resource management middleware the TLAM restricts reflective capability
 - Two levels
 - Base-level state represented and manipulated
 - Communication and scheduling observed, but not controlled
 - Base-level not aware of meta-level
 - TLAM uses a special purpose (ad hoc) rewriting mechanism to express the interaction of the base- and meta-levels.
-

The TLAM in Rewriting Logic

- TLAM and Maude
- Using Maude to model-check specifications of middleware services

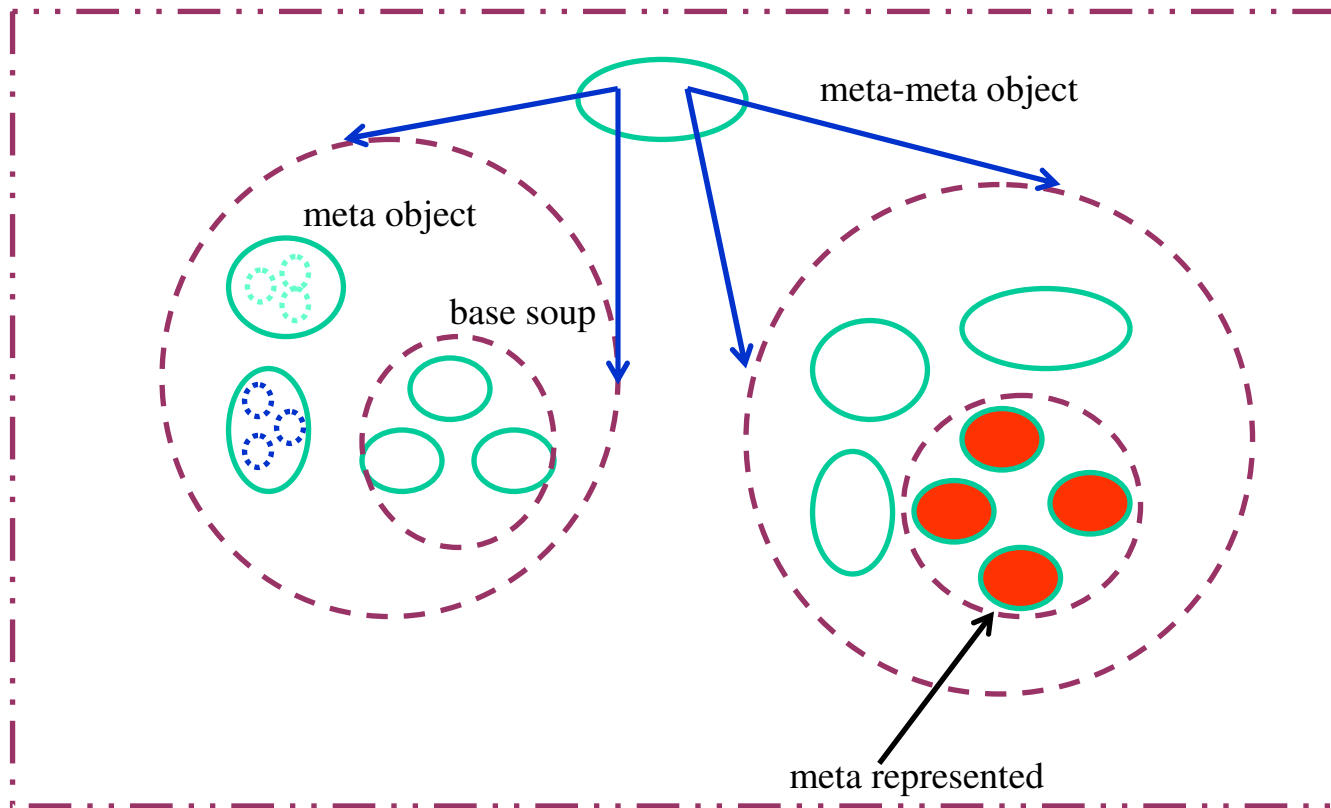


A Generalized Formal Model of Distributed Object Computation

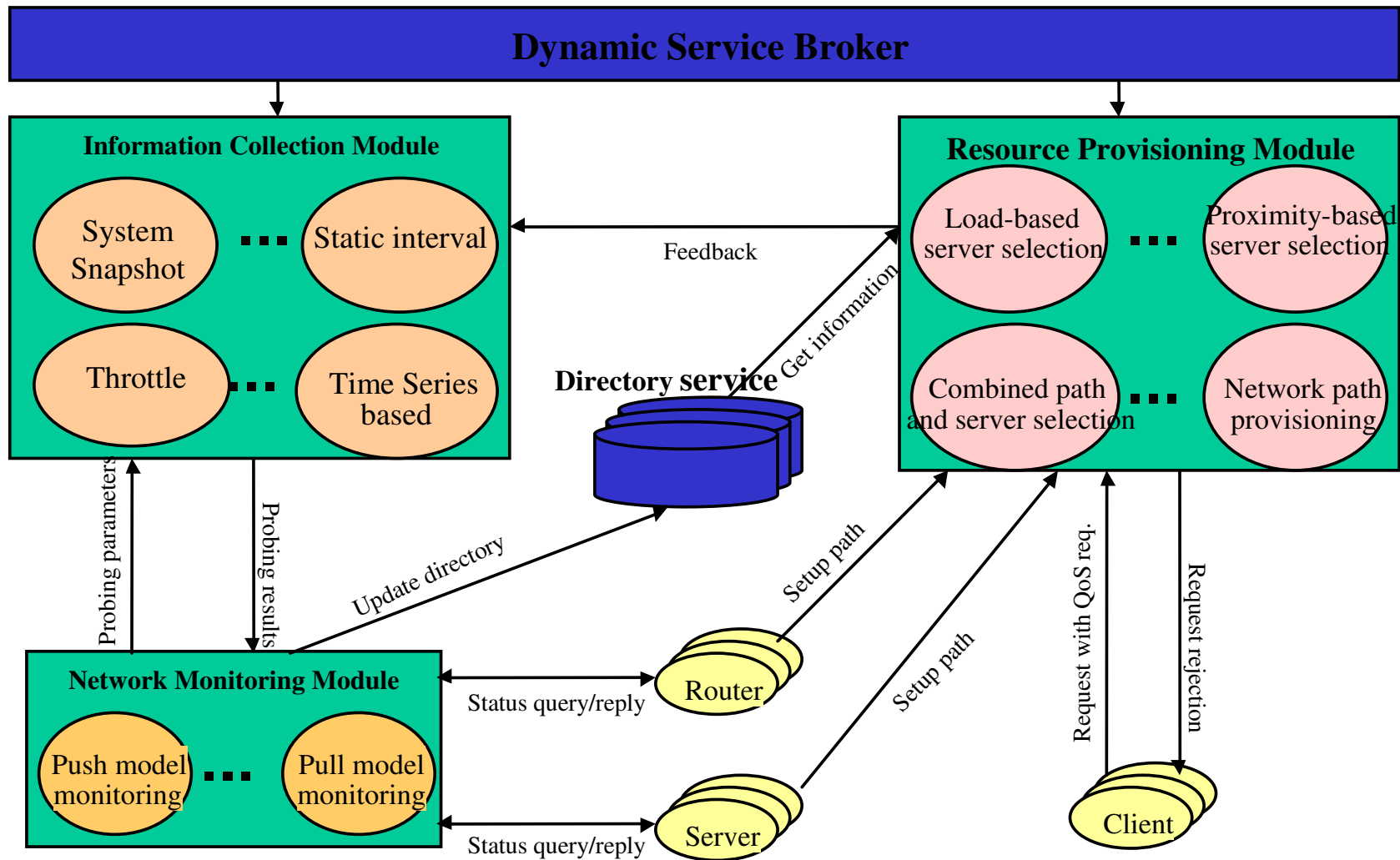
Using the ideas and reflective nature of rewriting logic a system is modeled as a soup of `reflective' objects.

A reflective object is a composite object consisting of a soup of meta-objects and a nested soup of base-objects whose structure and behavior is reflected in the meta-object soup. The meta-objects can be given full control of the base objects by using the meta-representation of the base-level provided by the logic.

Reflective System



AutoSeC Framework



Plan of work

- Formal specification of directory services
 - Stochastic and probabilistic notions of correctness in the TLAM
 - Interaction properties and constraints for metalevel services
 - Mobility management
 - Security – authentication and access control
 - Fault tolerance
 - Specification and verification using Mobile Maude
-