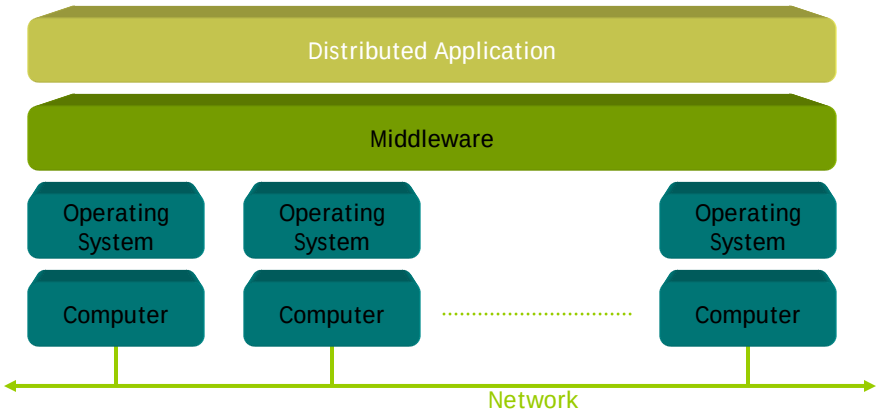
Episode 11: Middleware

Hannes Frey and Peter Sturm
University of Trier

Middleware

- Supporting distributed applications

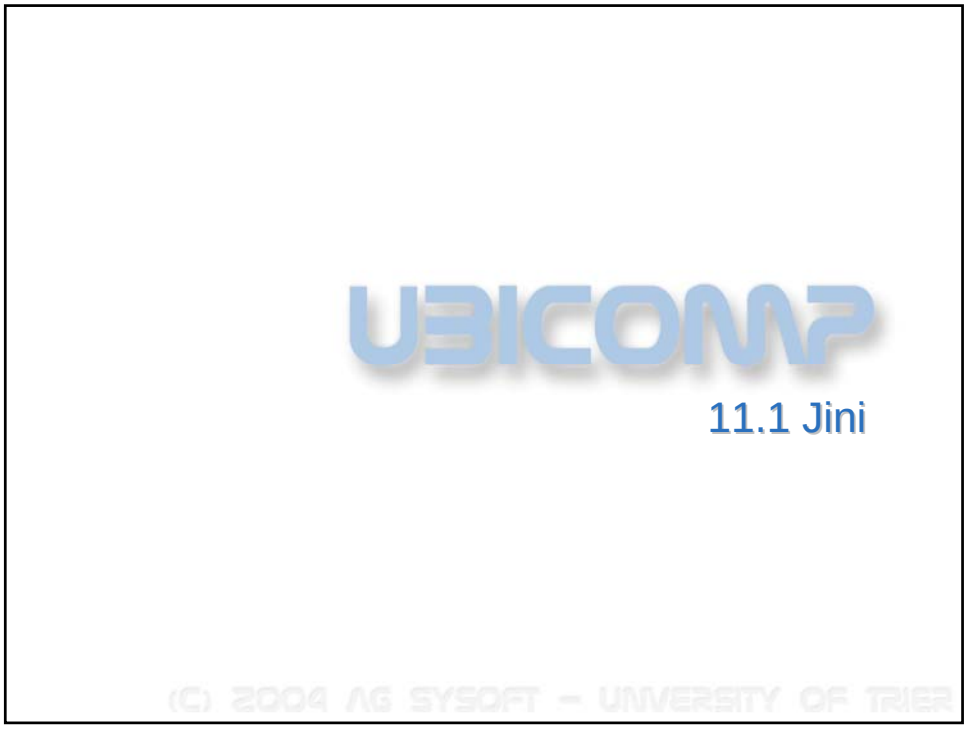


Issues

- Ease interaction and cooperation
  - Simple but strong abstractions for communication
  - “Brokering” of clients and servers, resp. peers
  - Transparency of some distributed properties
- Support heterogeneity
  - Marshalling of data
- Portability
  - Support for different hardware, operating systems, and (sometimes also) language environments

Common Middleware Systems

- Language dependend
  - Java, Java RMI
- Language independed
  - PVM, MPI
  - COM+, .NET
  - CORBA
- In most cases, adaptions to ubiquitous computing are investigated
  - Jini (Java-based approach)
  - Research projects in the context of .NET and CORBA



Jini

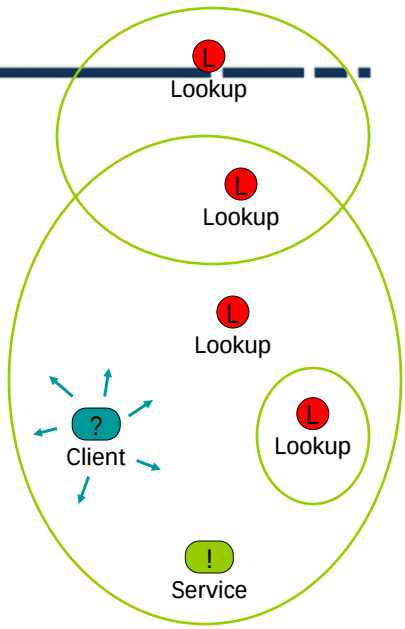
- Java-based software system
- Distributed system
  - Federates groups of users and required resources
  - Flexible and easy to administer network
- Jini consists of
  - Components providing the infrastructure for federating services in a distributed system
  - Programming model
  - Services which offer functionality to any member of a federation
- Jini is network agnostic
  - Physical connection has to be initiated before

Key Concepts

- Java based standard for spontaneous networking
- Key concepts
  - Discovery — Where are Jini Services (Communities)?
  - Lookup — What Jini-Services are there?
    - Service registration
    - Loadable Proxies
  - Leasing — Self-healing, Robustness
  - Remote Events — Asynchronity
  - Transactions — Integrity

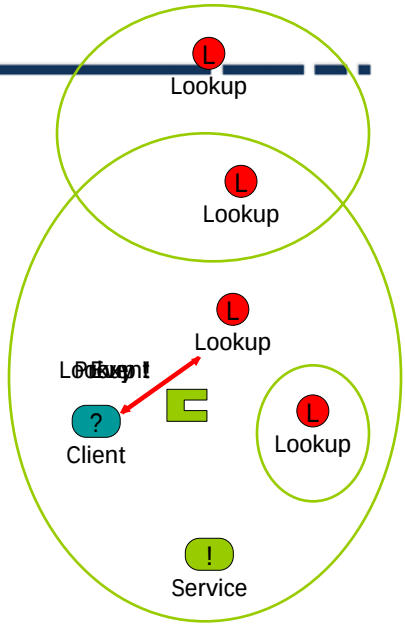
Discovery

- Federations
  - Set of clients using the same group of lookup servers
- Lookup Server = Directory service
- Possible protocols
  - Multicast Request
    - E.g. DHCP request
  - Multicast Announcement (e.g. about new Lookup Servers)
  - Unicast Discovery (known Lookup Server)
    - Access by URL



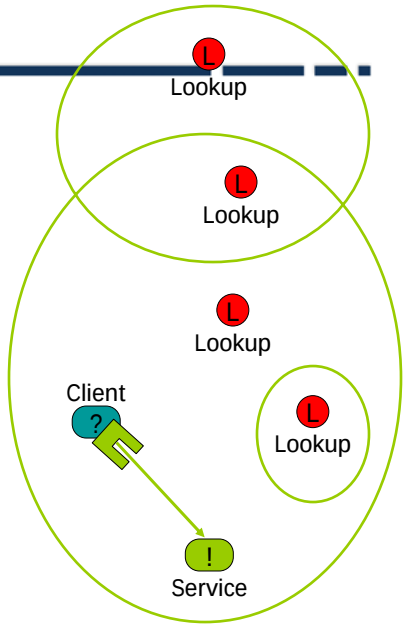
Lookup

- Events inform about lookup server
  - Client = DiscoveryListener
  - Asynchronous communication
  - no blocking wait
- Return value: Reference to proxy object
  - Different implementations possible
    - Simple hashing
    - LDAP, NIS, ...
  - Legacy protocol also possible
- Lookup server first returns proxy implements lookup service interface



Service Call

- Querying lookup server via type of interface
  - "Who implements interface X?"
  - Interface must be known by client
  - Implementation is downloaded dynamically
- Access via proxy object
  - Interpreted Java Bytecode
- Again, legacy protocols possible
  - Integration of very small devices
  - Wrapping of old software
- Preprocessing of request in proxy

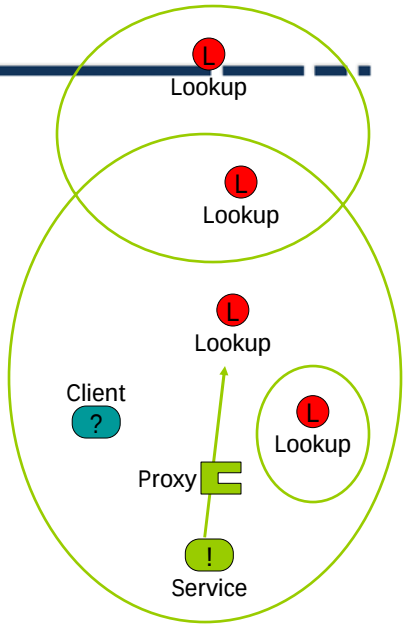


Proxy Objects

- Proxy = Any serializable object
- Code will be downloaded automatically if it is not available at client side
- Ease of administration
  - No driver support at client side required
  - No software installation at client side required
- Alternatives
  - Proxy itself implements service (simple services)
  - RMI stub (connects to remote server)
  - Legacy stubs with proprietary protocol

Leases

- Example service registration
- Delivering proxy object to lookup server
- Lease = Limits services for a given client with respect to time
  - Proposal for a deadline by service
  - Compared to limits and defaults within lookup server
  - Timely renewal of lease required
- Self-healing
- 3rd party leasing



Jini Runtime System

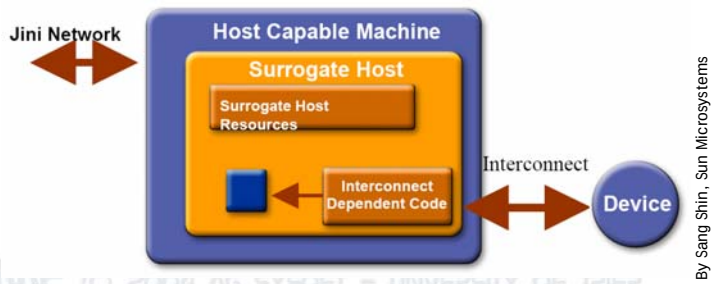
- Required
  - HTTP server (to export class files)
  - RMID (RMI activation daemon)
  - Lookup service (e.g. Reggie)
  - Security policy files
- Optional system services
  - Transaction manager (Mahalo)
  - JavaSpaces (Outrigger)
  - Lookup discovery service (Fiddler)
  - Lease renewal service (Norm)
  - Event mailbox service (Mercury)

JavaSpaces

- Distributed object repository
  - Persistence
  - Template matching lookup
  - Transactions
- Basic operations
  - Write: Put an entry into tupel space
  - Read: Return a matching entry from tupel space
  - Take: Remove a matching entry
  - Notify: Send an event if matching entry is written

Surrogate Architecture

- Jini addresses limited devices that
  - Run a limited JVM only (e.g. J2ME)
  - Run no JVM at all
  - …
- Such devices (and services) may participate in federations with the aid of surrogate hosts



Resources

- See www.jini.org
- Getting started
  - http://www.javapassion.com/jini/index.html

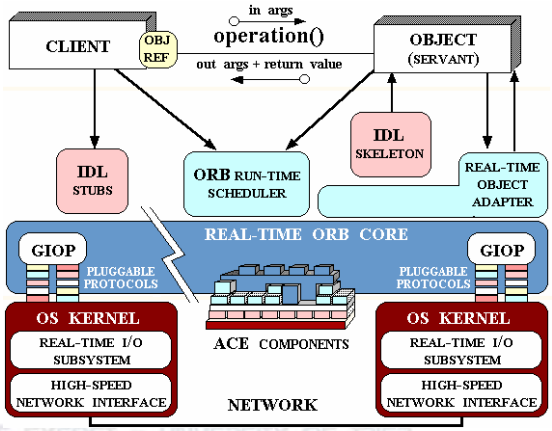
11.2 CORBA based Approaches

UbiComp and CORBA

- CORBA is no promising plattform in the first place
  - Heavy-weight
  - Requires compilation of interface definitions
  - Questionable performance over unreliable wireless communication
  - Ad-hoc networks are no target
- Reflective extensions
  - DynamicTAO
  - OpenCORBA

TAO

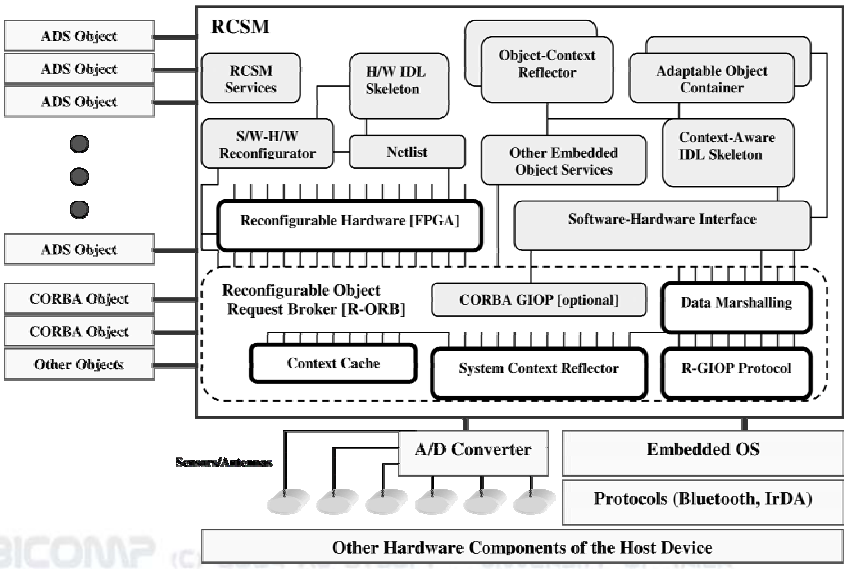
- University of California, Irvine, Douglas C. Schmidt
- Real-time implementation of CORBA with end-to-end quality of service
- DynamicTAO
  - Reflective version



RCSM

- S.S. Yau, F. Karim, Arizona State University, Tempe

RCSM Architecture



STEAM

- V. Cahill et al., Trinity College, Dublin
- STEAM = Scalable Timed Events and Mobility
- Event models
  - Peer to peer
  - Mediator
  - Implicit (for ad hoc networks)
- Proximity groups
- Event filters
  - Subject
  - Proximity
  - Content

Summary

- Most work in single hop networks
  - Assuming only low device mobility
  - Minor changes to existing middleware approaches prefered
- Multi hop networks are still a true research area
- Event-based communication model
  - One-to-One and One-to-Many possible
  - Fits into the required asynchronous world of ubiquitous computing
  - Avoids centralized control
  - Less tightly coupled
- Reflection is the key concept for future middleware