



MSc Course in Information and
Communications Technologies
Session 5: Middleware

Middleware



○ Overview of Lecture

- Problems of heterogeneity
- Open distributed processing and middleware
- Styles of middleware
- Focus on distributed object technology
- From objects to components

○ Additional reading

- Blair and Stefani, chapters 1 and 2; Emmerich, chapters 3, 4 and 5

The Crucial Problem of Heterogeneity

- Modern systems are inevitably heterogeneous
 - Hardware, operating systems, programming languages, etc.
- Emergence of open distributed processing
 - An open distributed system is one that conforms to well-defined interfaces and structures
 - Importance of standards



Benefits of Open Distributed Processing

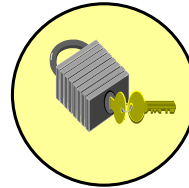
○ Portability

- An application or service developed for one platform (e.g. Windows) should be able to be ported with ease to another platform (e.g. Linux)

○ Interoperability

- An application should readily be able to interact with a service implemented on an unknown platform

The Role of Middleware



*Distributed Applications
and Services*

Platform Independent

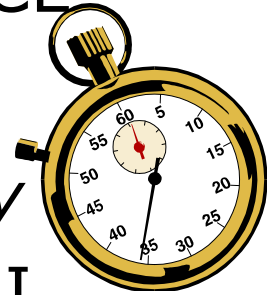
MIDDLEWARE

Platform Dependent



A History of Middleware

- First generation middleware
 - Based exclusively on the *client-server model*, and approaches such as 2- or 3- tier architectures
 - Examples include the Open Group's DCE
- Second generation middleware
 - Based on *distributed object technology*
 - Examples include CORBA and Java RMI
- Third generation middleware?
 - Based on emerging *component technology*, with more general n-tier architectures



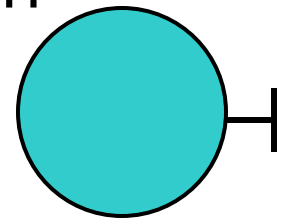
Other Styles of Middleware

- Message-oriented middleware
 - Event service (cf. publish-subscribe paradigm)
 - Examples include IBM's MQSeries and Sun's Java Message Service (JMS)
- Transactional middleware
 - Focus on reliability through atomic transactions
 - Examples include IBM CICS and Encina, BEA's Tuxedo and Microsoft's MTS
- Also database-oriented middleware, web services and grid middleware, etc....



Focus on Distributed Object Technology

- What is an object?
 - An object is an *encapsulation* of data and methods that manipulate the data
 - Objects are accessible only through their *interface* (the collection of methods defined on that object)
 - Interfaces are generally described using an *Interface Definition Language* (or IDL)



Example IDL (from CORBA)

```
interface inventory
{
    // attributes and type definitions
    const long MAX_STRING = 30;
    typedef long part_num; typedef long part_price;
    typedef long part_quantity;
    typedef string  part_name<MAX_STRING+1>;
    struct part_stock {
        part_quantity max_threshold;
        part_quantity min_threshold;
        part_quantity actual;
    };
    // operations
    boolean is_part_available (in part_num number);
    void get_price (in part_num number, out part_price price);
    void get_stock (in part_num number, out part_quantity quantity);
    long order_part (in part_num number, inout part_quantity quantity, in account_num account);
};
```



Rationale for Distributed Objects

- Application developers only need to know the interface of an object and are not concerned with the implementation details (*abstraction*)
- Crucially they do not care about the language of implementation or the platform supporting the object – they only see the interface (*language and platform independence*)
- One object can be replaced by another object as long as it offers the same interface, encouraging system *evolution*
- New objects can readily be introduced into the environment encouraging *extensibility*



Support for Legacy Systems

- What is a legacy system?
 - An established piece of software offering a key service for an organisation
 - Typically not integrated into the distributed environment and too complex to rewrite
- The solution
 - Develop an *object wrapper* that makes the service available in the (open) distributed environment

Distributed Objects: Key Players

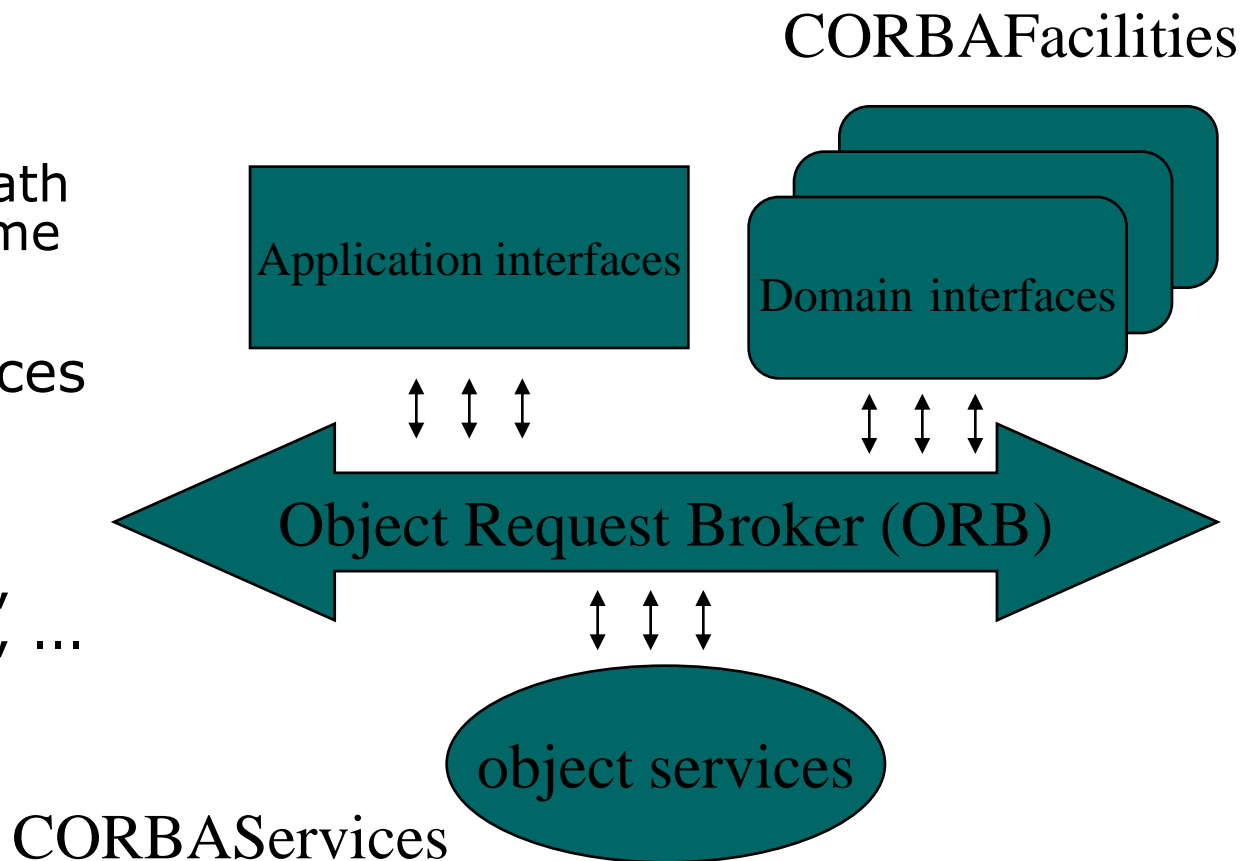
- Microsoft DCOM
- Java RMI
- OMG Common Object Request Broker Architecture (CORBA)
- ISO/ ITU-T Reference Model for Open Distributed Processing (RM-ODP)
 - c.f. ISO OSI

Focus on OMG CORBA

- What is the OMG (Object Management Group?)
 - A non-profit organisation with over 800 member organisations
 - Promotion of the use of object technology in distributed systems (CORBA, UML, etc)
- Goals of CORBA
 - The definition of a complete object-based middleware architecture

The CORBA Architecture

- Example domains
 - Financial services, health care, real-time systems, ...
- Example services
 - Naming service, security, transactions, time service, ...

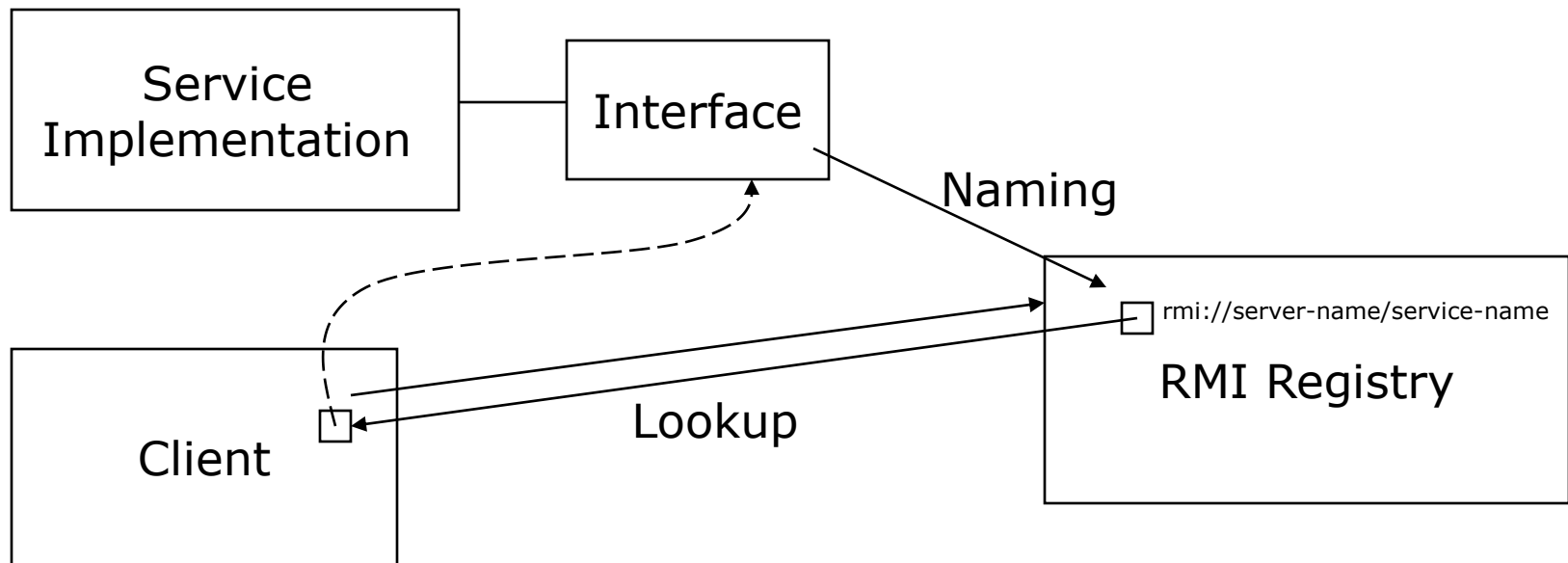


The Demise of CORBA

- Problems with design and incomplete implementations
- Lack of clear goals
- Etc
 - <http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=396>

Focus on Java RMI

- Sun's distributed Object technology for Java
- Similar Design to Corba



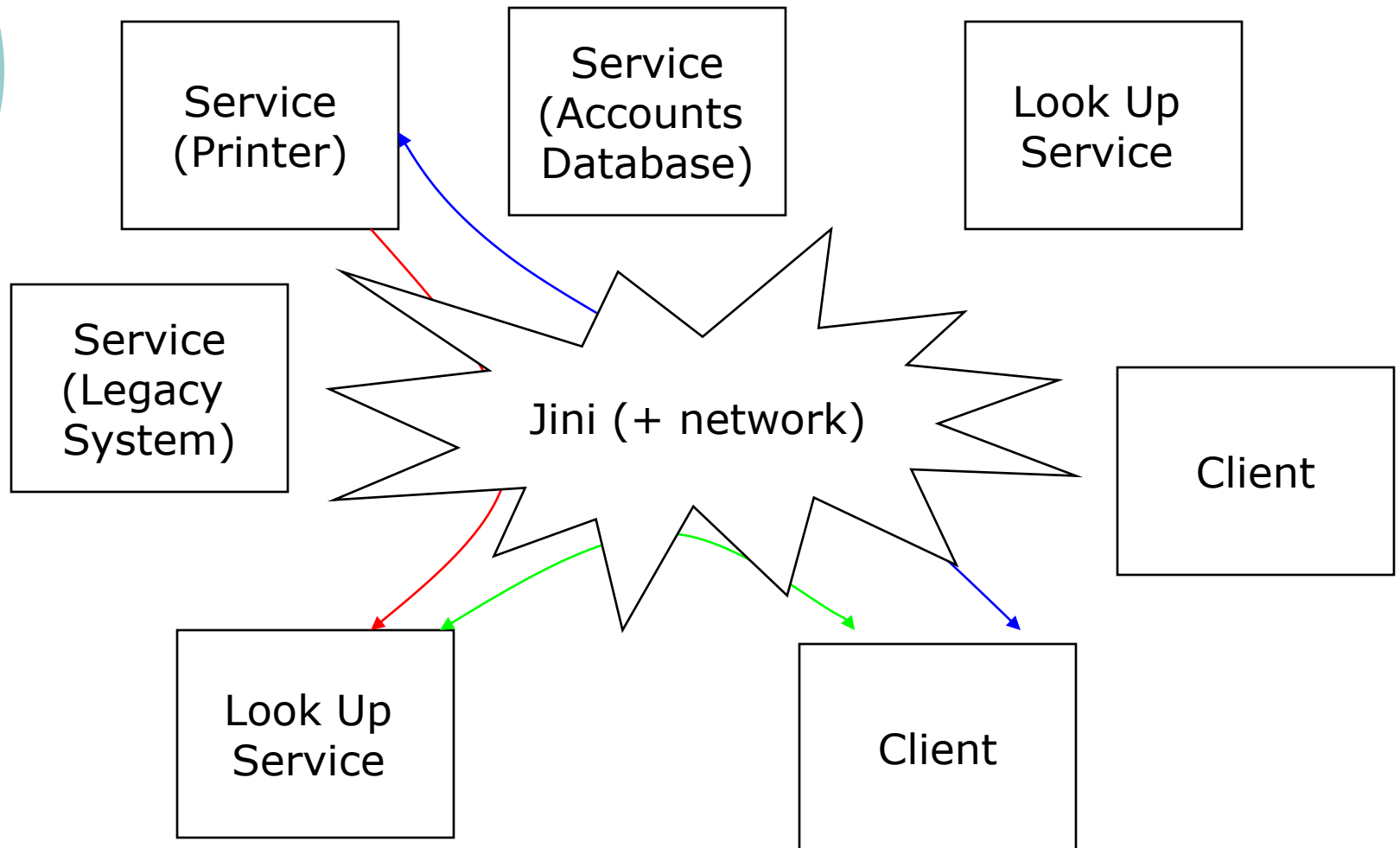
- Lets look at some code again...



Brief Overview of Jini

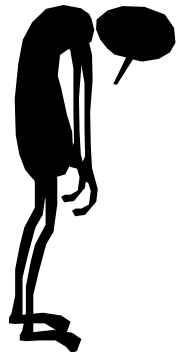
- Developed by Sun
 - Java Technology
 - Uses RMI for Communication
- Provides Access to distributed services
 - Includes a service discovery mechanism etc.
- Scalable, Dynamic, Secure, Resilient, Reliable

A Jini Federation



Assessment

- Distributed objects provide both language and platform independence, hence solving many of the problems of heterogeneity
- Distributed objects can be used to integrate legacy systems into modern distributed applications
- Applications are still complex to develop and deploy, e.g. w.r.t. calling services

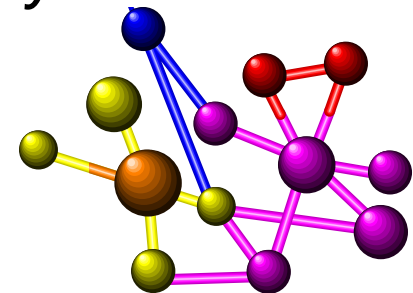


From Objects to Components

- What is a component [Szyperski]?

“ a unit of *composition* with *contractually specified interfaces* and *explicit context dependencies* only”

“in this context, a component can be *deployed independently* and is subject to *third-party composition*”



Rationale for Components



- Time to market
 - Improved productivity/ reduced complexity
 - Emphasis on reuse
 - The emergence of a component marketplace
- Programming by assembly (manufacturing) rather than development (engineering)
 - Reduced skills requirement

Packaging and Deployment

○ Packaging

- To ship a component may require many separate files (component code and meta-data)
- Solution is to package them into a standardised archive format
- Potential use of XML

○ Deployment

- Support typically provided for the installation, configuration, server-side management, etc

Components: Key Players



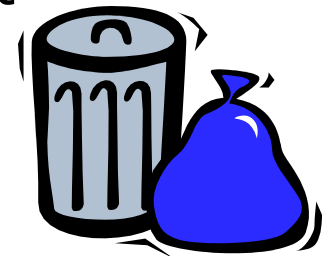
- Microsoft and components
 - Development of COM/ DCOM, COM+ and, more recently, .NET
- SUN and components
 - Development of Java Beans and EJB
- The OMG and components
 - Emerging CORBA v3 standard featuring the CORBA Component Model (CCM)

A Brief Look at Enterprise Java Beans

“The Enterprise JavaBeans architecture is a component architecture for the development and deployment of component-based distributed business applications. Applications written using EJB are scalable, transactional, and multi-user secure. These applications may be written once, and then deployed on any server platform that supports the EJB specification.”

The Key Concept of a Container

- What is a container?
 - Components are placed into a container, with this container offering *implicit* distributed systems management
 - Security
 - Reliability
- How is this achieved?
 - Containers intercept incoming requests and carry out necessary steps for security etc
 - Completely hidden from the component



Other Key Concepts



- Support for packaging
 - A standard format is defined for wrapping components (beans) together with necessary information about these components (meta-data)
 - Known as JAR files (cf. zip files, etc)
 - Can be used for transmission across the Internet
- Support for deployment
 - Tools provide for taking JAR files and automatically installing the resultant components in the distributed environment



Expected Learning Outcomes

- You should have a strong understanding of the role and motivation for middleware, and also how it solves problems of heterogeneity
- You should be able to discuss the rationale for distributed object technology
- You should be able to outline the basic concepts of CORBA including the key roles of the ORB and object services
- You should be able to capture the essence of the component-based approach to distributed system development and also how this differs from previous distributed object technologies
- You should be aware of the range of component-based technologies currently available, and also be able to describe the main features of the EJB architecture