M1 L2
Distributed Object Architecture

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Overview

- **Distributed Object Architecture**
- **CORBA**
- **Enterprise Service Bus**

Distributed Software Architectures

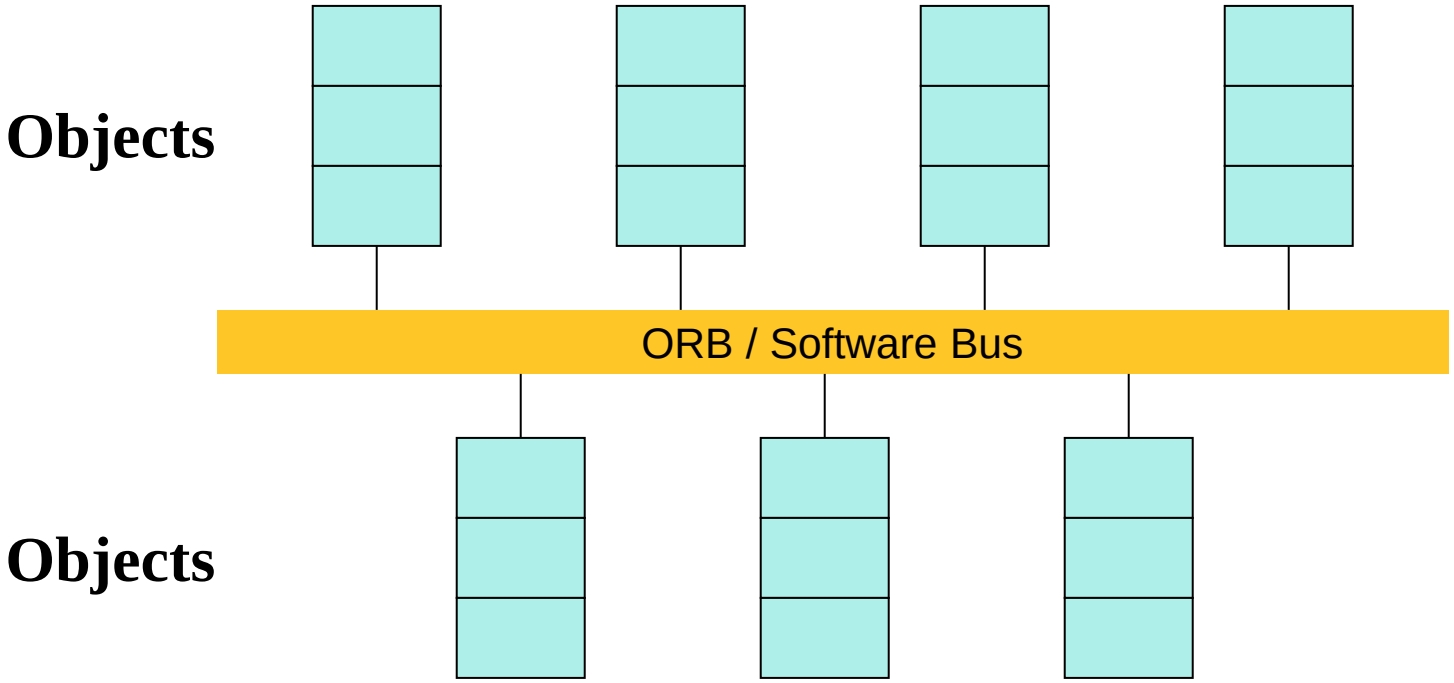
- Tiered architecture: Client and server are different in their roles
- Distributed Object Architecture
  - No distinction between clients and servers.
  - Any object in the system may provide and use services from other objects

Distributed Object Architecture

In distributed object architecture:

- There is no distinction between clients and servers
- Each distributable component is an object that provides services to other objects and receives services from other objects
- Object communication is through a middleware system called an object request broker (ORB, also called software bus)
- However, more complex to design than client-server systems
- No explicit separation of duties – The same person/team designs the same types of objects and the entire system, which requires complex skill to handle.
- There are competing standards.

Distributed Object Architectures



Advantages of Distributed Object Architecture

- It is an open system architecture that allows new resources to be added to it as required;
- The system is flexible and scalable;
- It is possible (e.g., written in the same language) to reconfigure the system dynamically with objects migrating across the network as required.
- It allows the system designer to delay decisions on where and how services should be provided;

Uses of Distributed Object Architecture

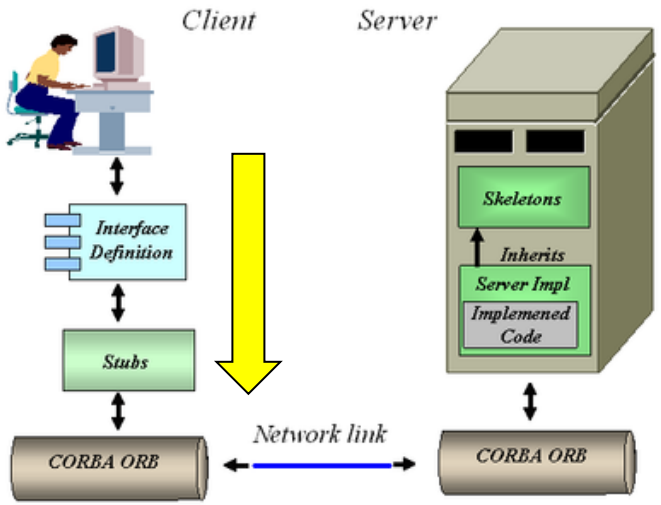
- **As a logical model that allows you to structure and organize the system.**
  - You focus on providing application functionality solely in terms of services and combinations of services;
- **As a more flexible approach to the implementation of client-server systems.**
  - Both clients and servers are realized as distributed objects communicating through a software bus.
  - The server does need to not know the client.

Object Request Broker (ORB)

- The ORB handles object communications.
- A service provider makes its services known via stubs.
- Using an ORB, the calling object binds an IDL stub that defines the interface of the called object.
- Calling this stub results in a call to the ORB, which then calls the required object through a published IDL (Interface Definition Language) skeleton that links the interface to the service implementation.

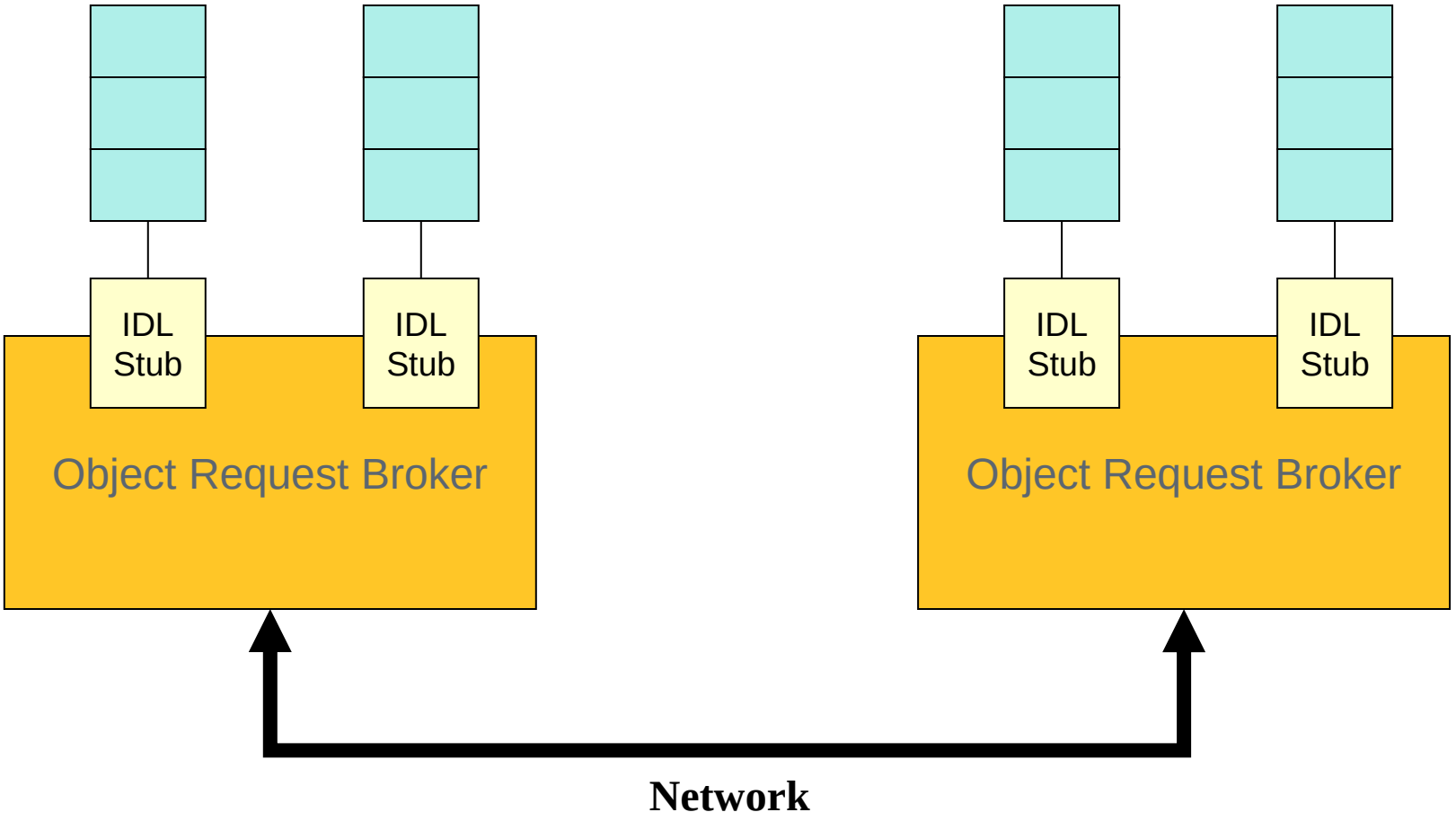
Publish / register

Agent / broker



ORB-based Object Communication

Distributed Object Architecture: Objects can use each other's services.

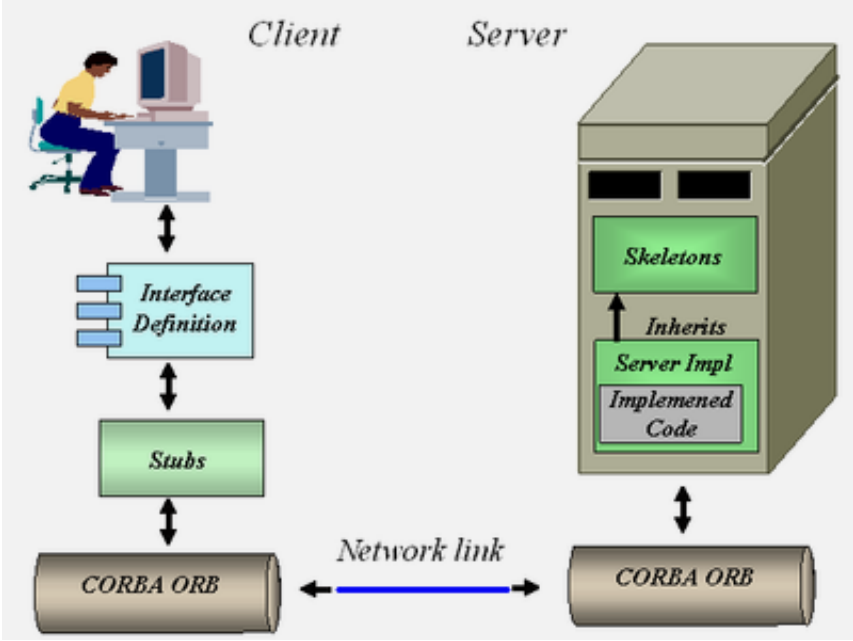


CORBA: Common Object Request Broker Architecture

- Defined by the Object Management Group (OMG)
- An international standard for an *Object Request Broker* - middleware to manage communications between distributed objects/components
- Enable software components written in multiple computer languages and running on multiple computers to interoperate;
- Several implementations of CORBA are available;
- DCOM was an alternative (competing) approach by Microsoft to object request brokers.

CORBA

- CORBA objects are comparable, in principle, to objects in C++ and Java;
- The objects have a separate interface defined by the Interface Definition Language (**IDL**) that is similar to C++;
- There is a mapping from this IDL to programming languages (C++, Java, etc.);
- Therefore, objects written in different languages can communicate with each other;
- CORBA can be used for implementing a Client-Server Architecture



MS DCOM: Distributed Component Object Model

- Originally, Microsoft distributed object system was called OLE (Object Linking and Embedding) and then “Network OLE”;
- It was extended to Microsoft's COM (Component Object Model) in 1993, which provided the communication capacity among objects;
- Windows 2000 significantly extended COM and renamed it COM+. Then, COM+ becomes DCOM later;
- DCOM allows software components to distribute across several networked computers to communicate with each other;
- All these technologies are integrated into and replaced by Visual Studio .NET, which supports both Object-Oriented and Service-Oriented Architecture (SOA).

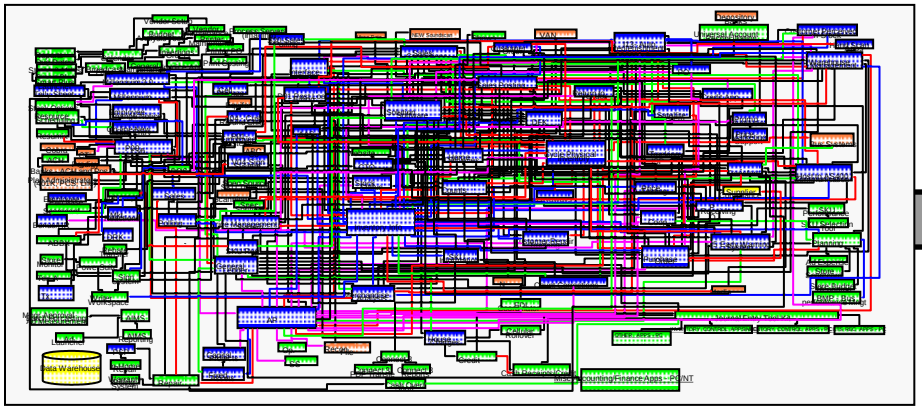
Enterprise Service Bus (ESB)

Shifting from Distributed Object Orientation to Service Orientation

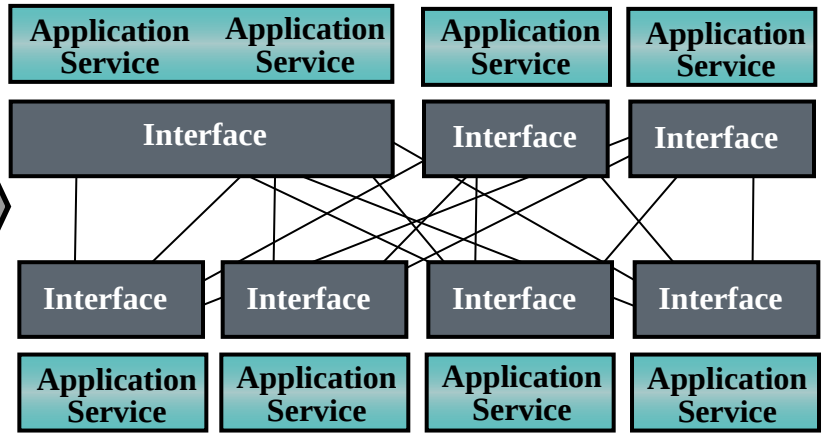
- An ESB provides an abstraction layer on top of an implementation of an enterprise messaging system;
- It allows integration architects to exploit the value of messaging without writing code.
- It is the service-oriented version of ORB.
- Contrary to the more classical Enterprise Application Integration (EAI) approach (tightly coupled), ESB provides a loosely coupled platform to support distributed functions, with distributed (independent) deployment where needed, while still working in harmony.

Modeling Business in SOA

Turn this model



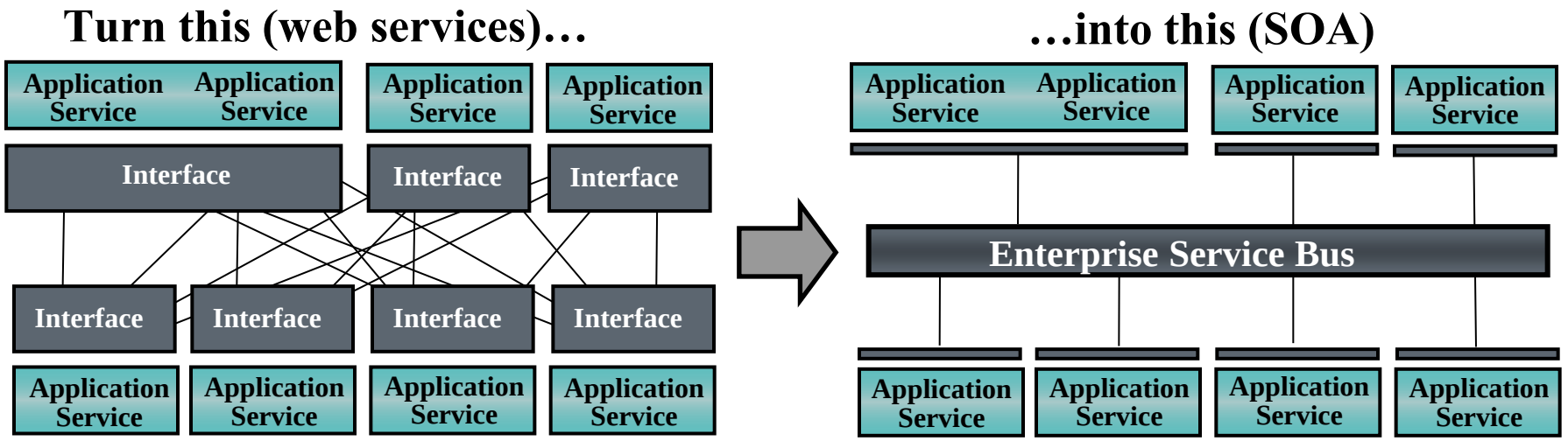
…into this (SOA).



- ✓ Rich business abstractions describe the application interface
- ✓ Decouples the interfaces from the business applications
- ✓ The number and complexity of the interfaces are reduced
- ✓ Business applications and their interfaces become reusable

Source: Carter Sandy Keynote Presentation, IBM, 2005

The ESB shrinks those interfaces further



- ✓ Decouples the point-to-point connections from the interfaces
- ✓ Allows for dynamic selection, substitution, and matching
- ✓ Enables more flexible coupling and decoupling of the applications
- ✓ Enables you to find both the applications and the interfaces for re-use

Source: Carter Sandy Keynote Presentation, IBM, 2005


Ira A. Fulton Schools of
Engineering
Arizona State University