Topics for today

- CORBA

CORBA (Common Object Request Broker Architecture)

A specification or standard which specifies how objects communicate with each other over a network.

It is a middleware based architecture

The ORB is the heart of the distributed component architecture

Allows developers to define distributed component architectures without worrying about the underlying network Communications and programming language

Motivation behind CORBA

Distributed applications cause a lot of problems

- Participating systems may be heterogeneous
- Access to remote services has to be location transparent
- Remote objects have to be found and activated
- State of objects has to be kept persistent and consistent
- Security has to be dealt with

so we want an architecture that…

- …supports a remote method invocation paradigm
- …provides location transparency
- …allows to add, exchange, or remove services dynamically
- …hides system details from the developer

The result is the CORBA architecture

What is CORBA?

- CORBA is a standard (not a product!)

- allows objects to transparently make requests and receive responses

- enables interoperability between different applications
- ⍰ on different machines
- ⍰ in heterogeneously distributed environments

CORBA design goals

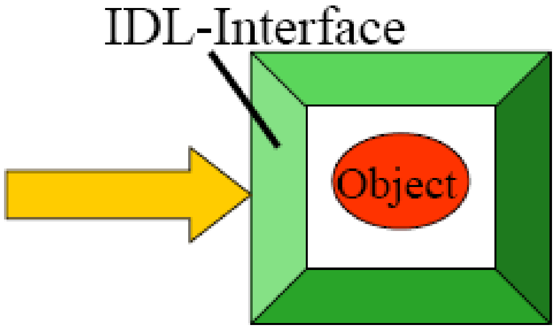
Independence of

- hardware platform
- programming language
- operating system
- specific Object Request Broker
- degree of object distribution

Open Architecture:

- Language, platform and location transparent
- Language-neutral Interface Definition Language (IDL)

CORBA works with interfaces



- all CORBA Objects are encapsulated

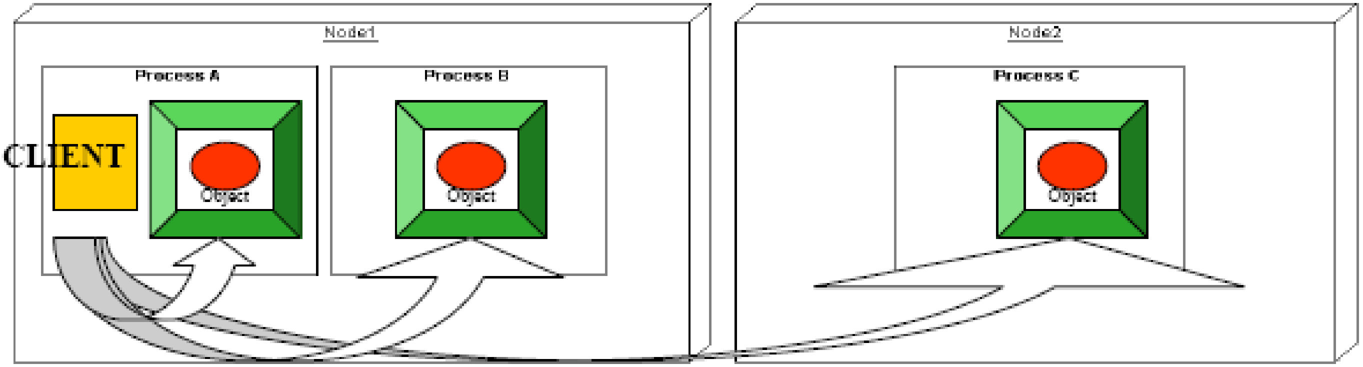
- Objects are accessible through interface only

- Separation of interfaces and implementation enables multiple implementations for one interface

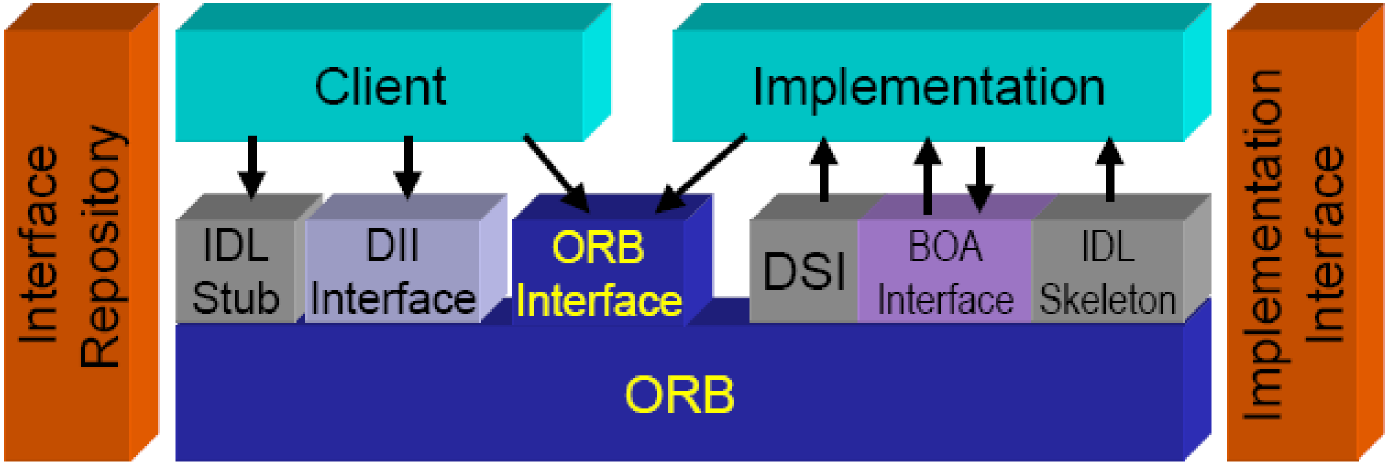
CORBA is *Location Transparent*

For the client it doesn't matter if the object he is operation on is running…

- on the same processor and even in the same process
- on the same processor, in a different process
- in different process on another processor



CORBA Architecture



ORB Core

CORBA ORB Interfaces

ORB Interface

ORB interface: contains functionality that might be required by clients or servers

DII Interface

Dynamic Invocation Interface(DII): used for dynamically invoking CORBA objects that were not known at implementation time

DSI

Dynamic Skeleton Interface(DSI): helps to implement generic CORBA servants

BOA Interface

Basic Object Adapter(BOA): API used by the servers to register their object implementations

Ways to achieve the goals of CORBA

- Interface Definition Language (IDL)
- Object Request Broker (ORB)
- Internet Inter-ORB protocol (IIOP)

Interface Definition Language (IDL)

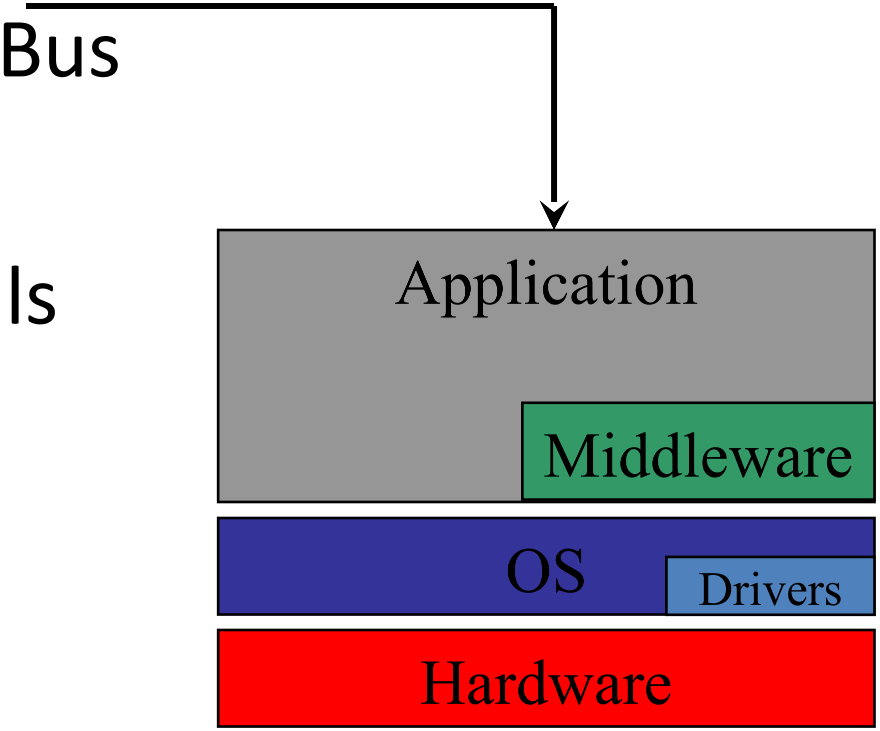
- Language Independence
- Defines Object Interfaces
- Hides underlying object implementation
- Language mappings exist for C, C++, Java, Cobol, Smalltalk, and Ada

Object Request Broker (ORB)

- What is it?
- ORB Architecture (CORBA architecture)

What is it?

- Implementation of CORBA specification
- Middleware product
- Conceptual Software Bus
- Hides location and implementation details about objects

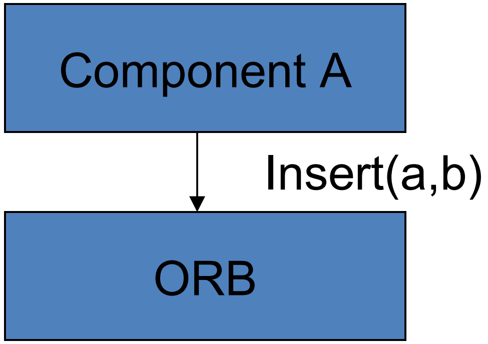


Two major task of ORB’s

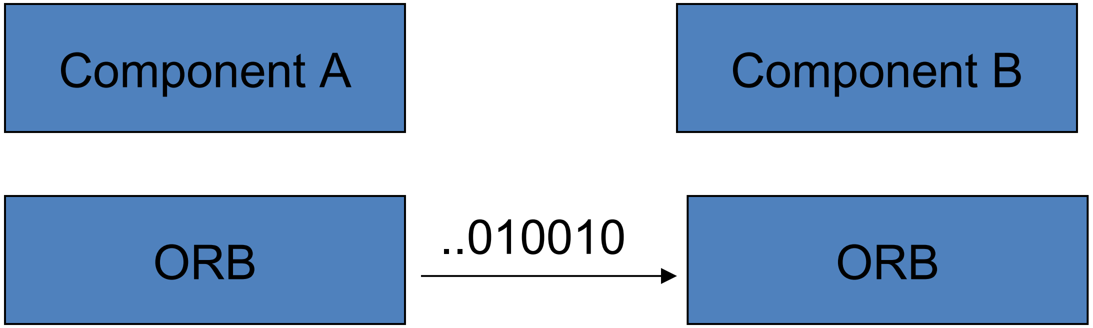
- The ORB locates a remote component, given its reference.
- The ORB also makes sure that the called method receives the parameters over the network correctly. This process is known as **Marshalling**. In reverse manner, the ORB also makes sure that the calling component receives back the return values, if any, from the called method. This process is known as **Un-Marshalling**.

Example of how the ORB's work

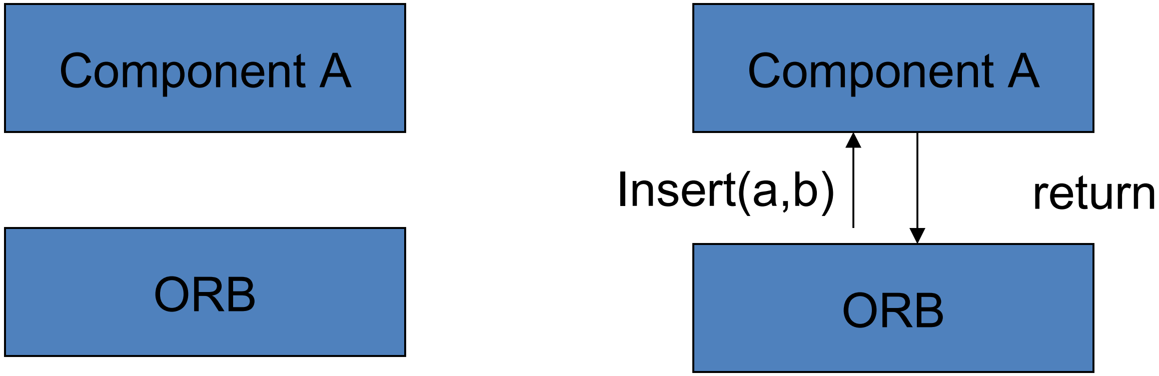
- Component A calls *Insert* method and passes two parameters, *a* and *b*



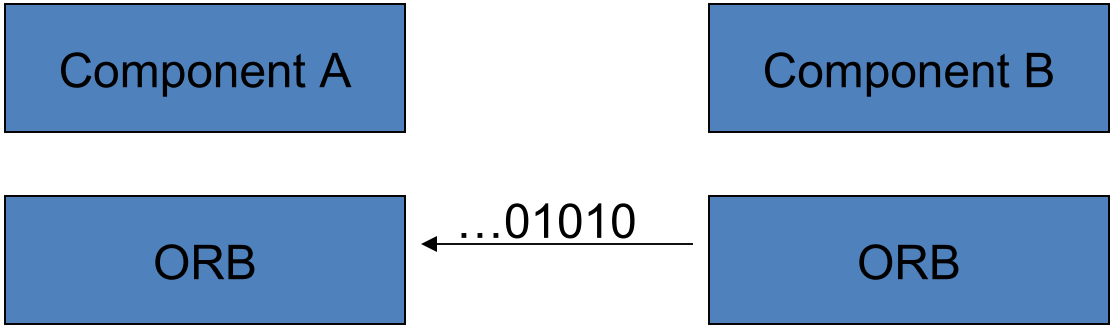
- Component A *ORB* picks up the request and realizes that *Insert* method is defined in component B and passes the method name and parameters in binary form(i.e marshals them) across the network to the ORB at the node where component B is located.



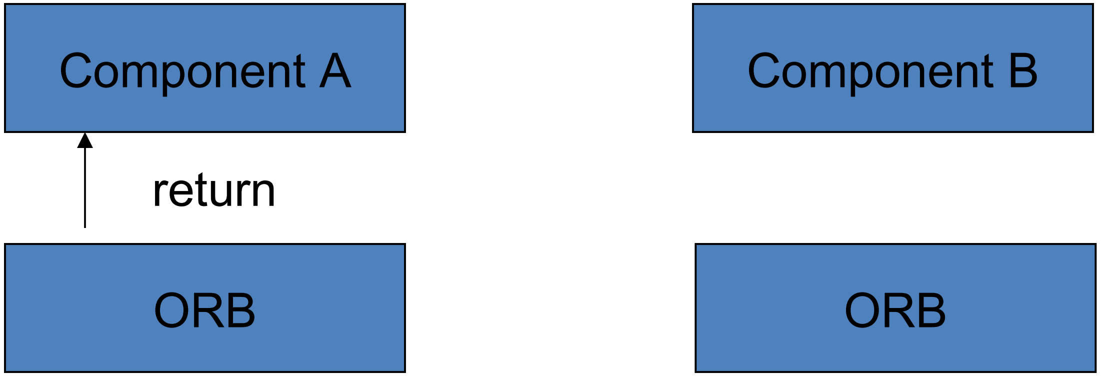
- ORB at component B’s end picks up this request and converts the binary data back into its original form. Calls *Insert* method with parameters, executed and result is returned to the ORB



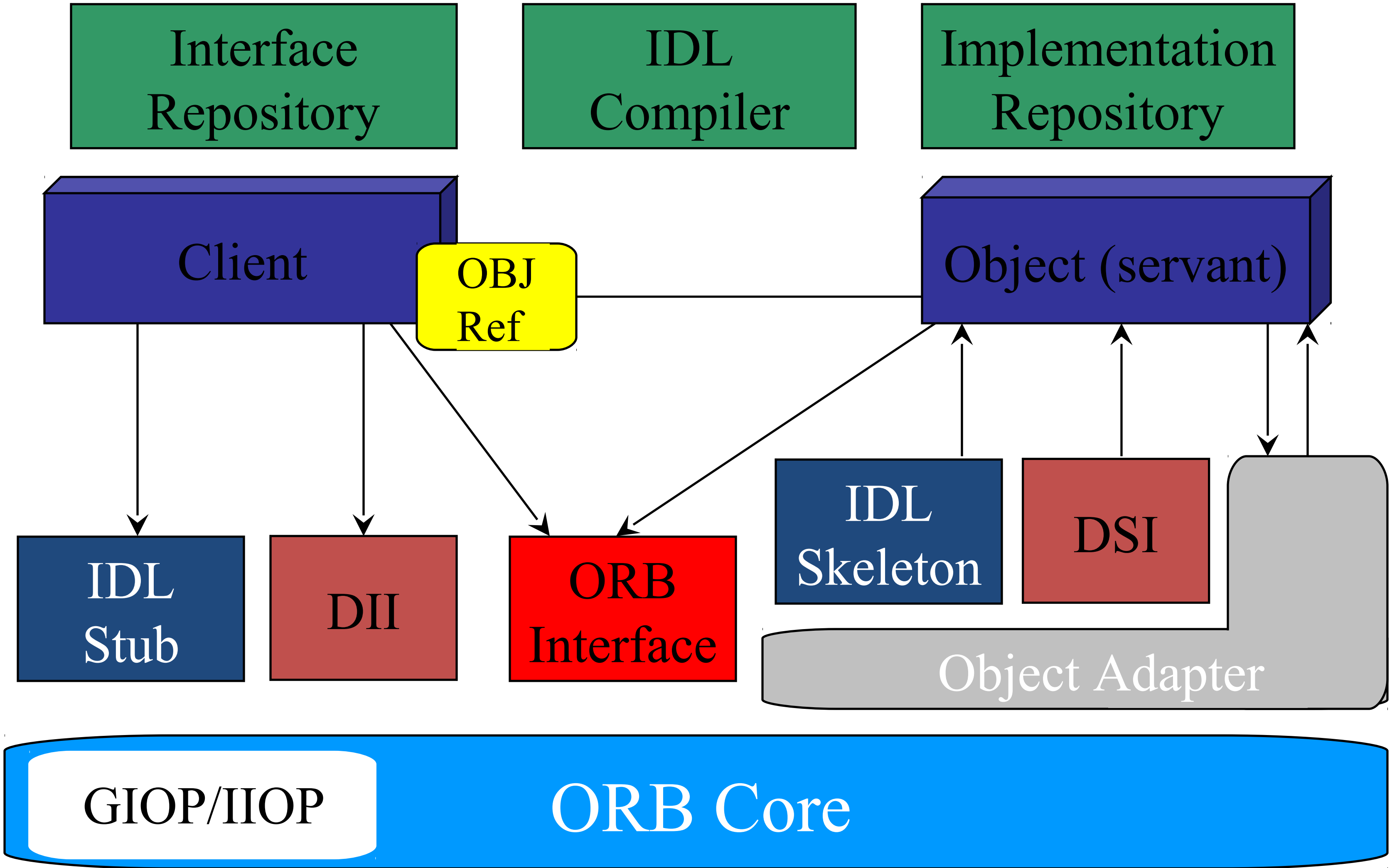
- Finally, the ORB at component B’s end takes this return value, converts it into binary equivalent and sends it across the network back to ORB at A’s component (i.e un-marshals the request)



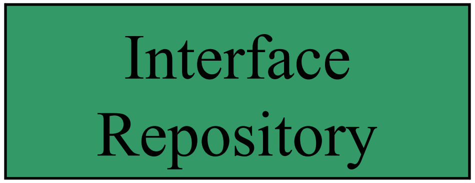
- The ORB at component A’s end picks up this binary data, converts back to the original values and gives it to component A



ORB Architecture



Interface Repository



Interface Repository

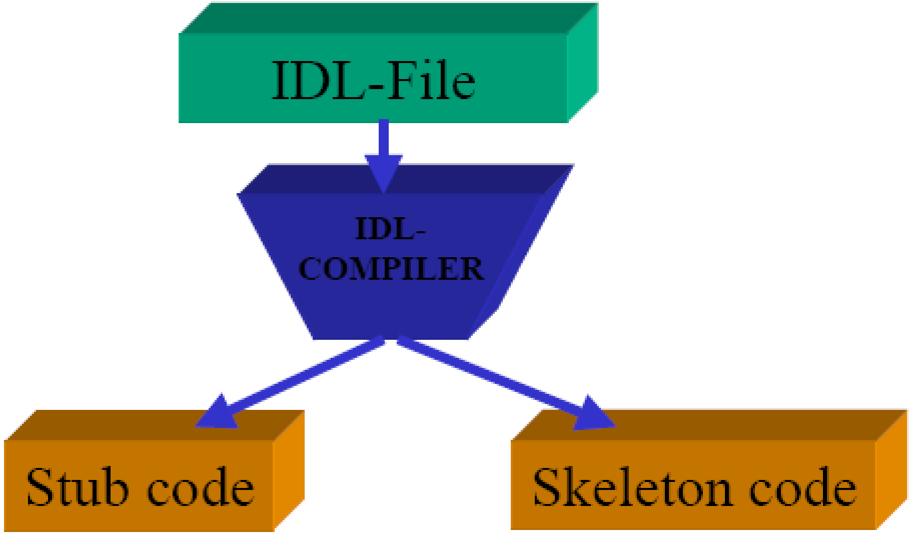
- Database of object definitions
- Contains metadata about each object
- Allows for introspection
- Allows clients to discover interfaces at run-time
- Used in support of dynamic invocations

IDL-Compiler

translates the IDL-specifications into

- *IDLstubs* (client-side)
- *IDL-skeletons* (server-side)

in the desired programming language



Implementation Repository

Implementation Repository

- Contains information that allows the ORB to locate and activate object implementations
- Provides information about the classes a server supports, the objects that are instantiated, and their IDs

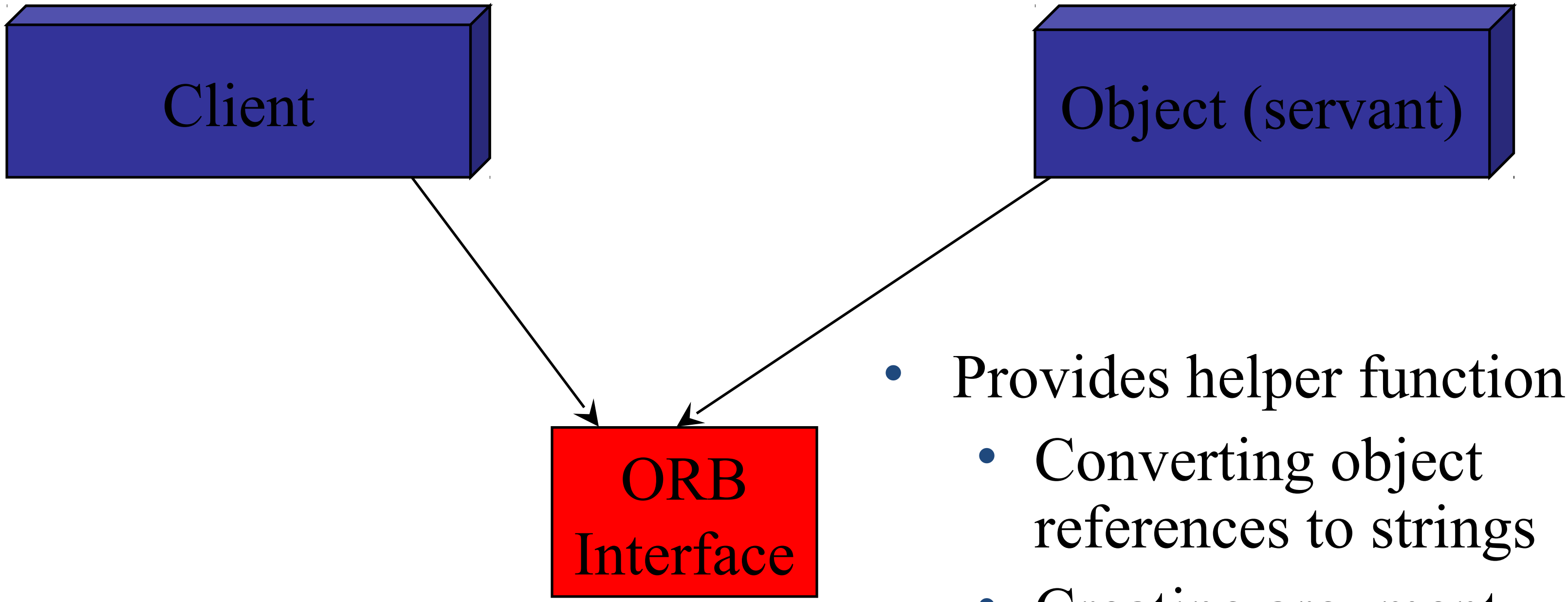
ORB Core

- Provides mechanism for transparently communicating client requests to target object implementations
- Makes client requests appear to be local procedure calls
- GIOP – General Inter-ORB Protocol
- IIOP – Internet Inter-ORB Protocol

GIOP/IOOP

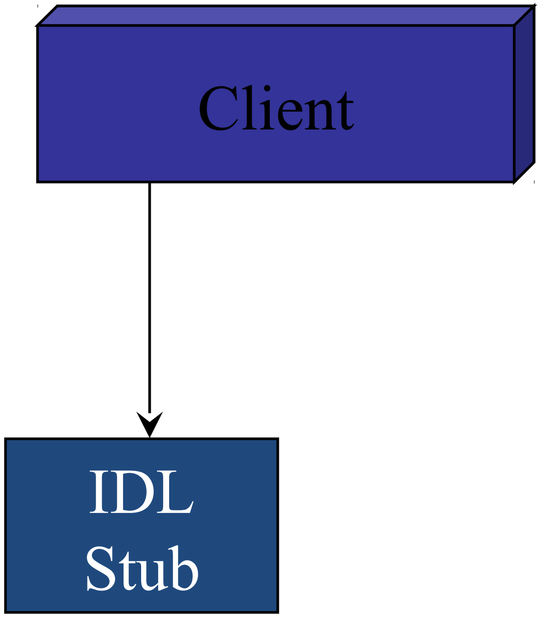
ORB Core

ORB Interface



- Provides helper functions
  - Converting object references to strings
  - Creating argument lists for requests made through DII

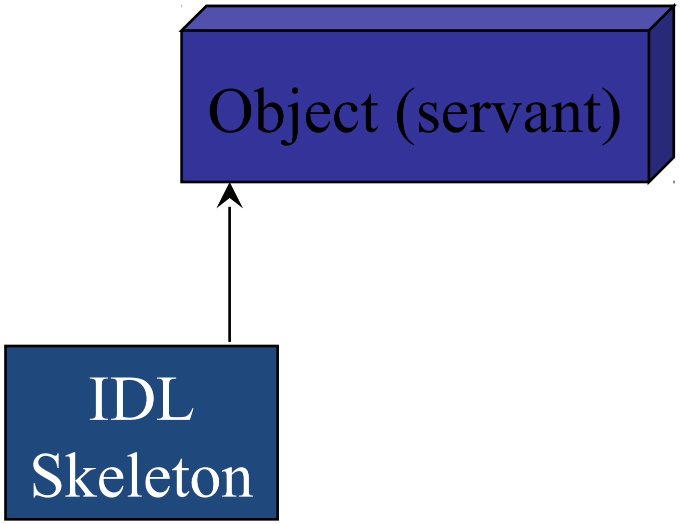
IDL Stub



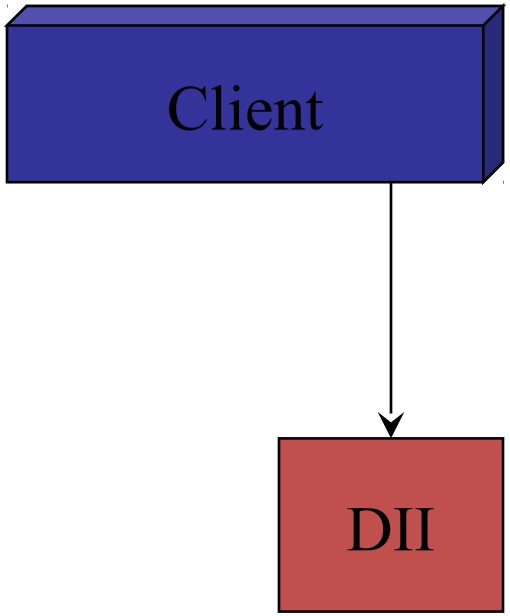
- Static invocation interface (SII)
- Marshals application data into a common packet-level representation
  - Network byte order (little-endian or big-endian)
  - Size of data types

IDL Skeleton

- Demarshals the packet-level representation back into typed data that is meaningful to an application
  - Network byte order (little-endian or big-endian)
  - Size of data types



Dynamic Invocation Interface



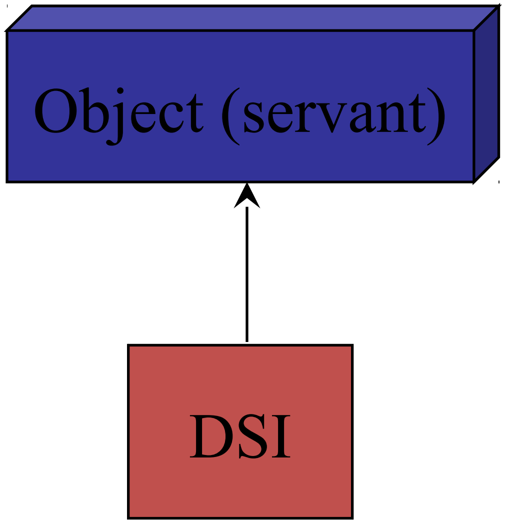
- Dynamically issue requests to objects without requiring IDL stubs to be linked in
- Clients discover interfaces at run-time and learn how to call them

Steps:

1. Obtain interface name
2. Obtain method description (from interface repository)
3. Create argument list
4. Create request
5. Invoke request

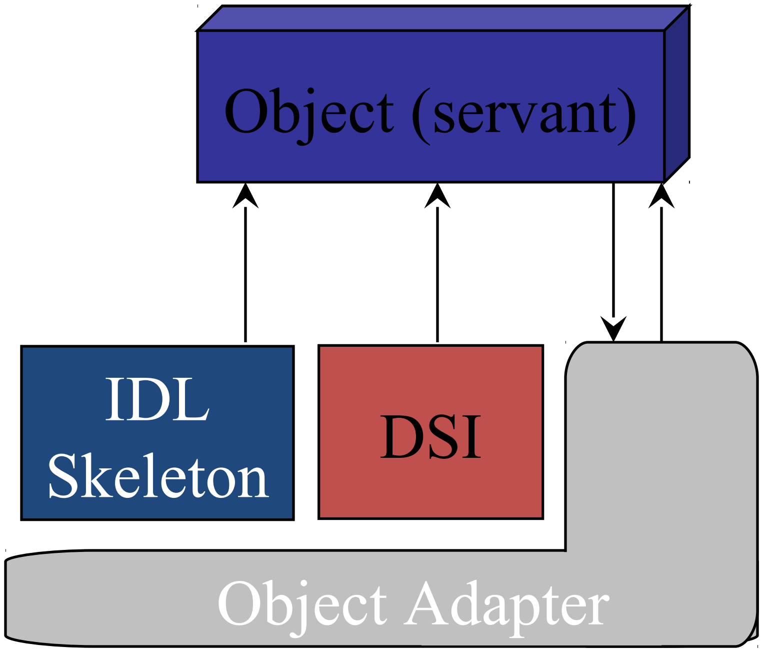
Dynamic Skeleton Interface

- Server side analogue to DII
- Allows an ORB to deliver requests to an object implementation that does not have compile-time knowledge of the type of object it is implementing

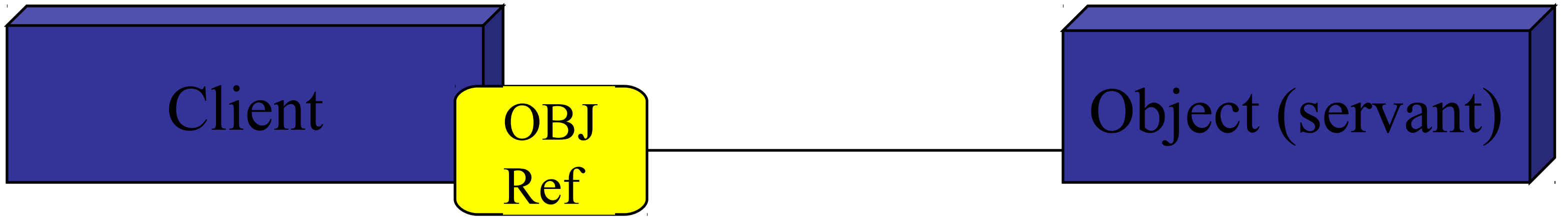


Object Adapter

- Accept requests for service on behalf of the server's objects
- Demultiplexes requests to the correct servant
- Dispatches the appropriate operation upcall on the servant
- Registers classes it supports and their run-time instances with the implementation repository

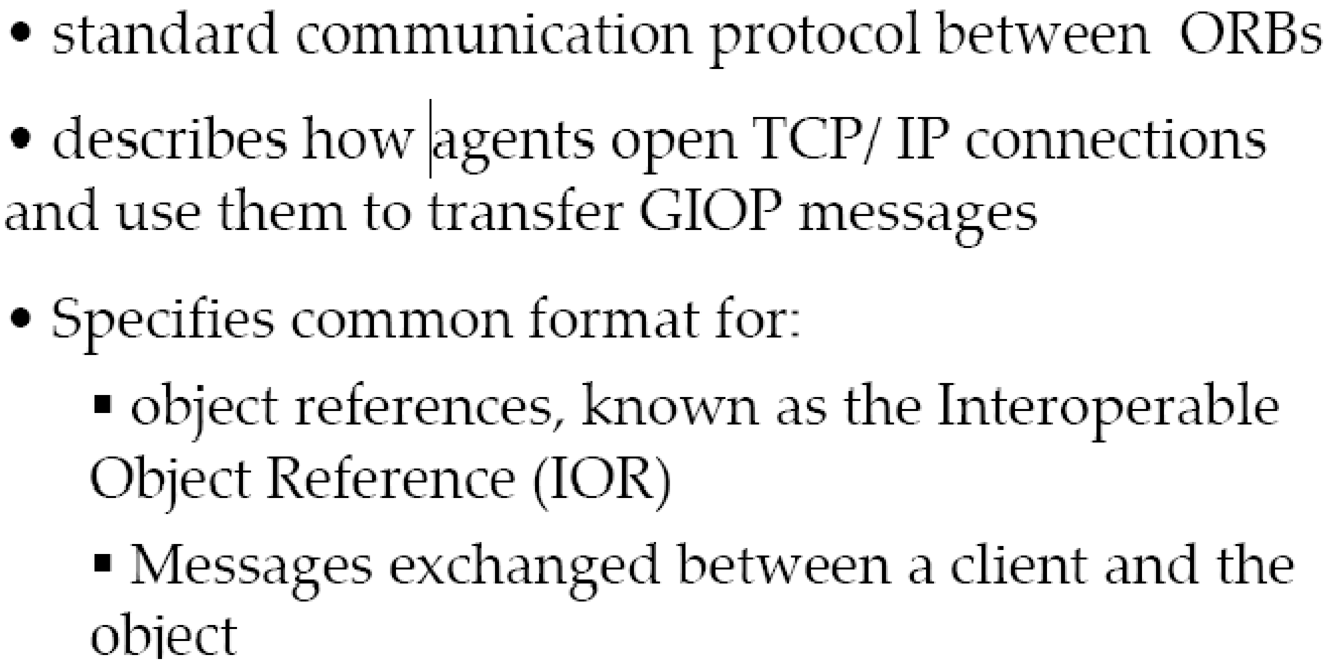


Object Reference



- Interoperable Object Reference (IOR)
  - Uniquely identifies each object
  - Contents
    - Type Name (repository ID)
    - Protocol and Address Details
    - Object Key (object adaptor name, object name)

IIOP - Internet Inter-ORB Protocol



- standard communication protocol between ORBs
- describes how agents open TCP/ IP connections and use them to transfer GIOP messages
- Specifies common format for:
  - object references, known as the Interoperable Object Reference (IOR)
  - Messages exchanged between a client and the object

GIOP-General Inter ORB Protocol

Advantages of CORBA

- CORBA allows methods on a remote object to be accessed as if they were on the local machine

- CORBA is a mature technology, support and tools are widely available

- Can deal with heterogeneous systems

- Legacy-systems can be integrated

DCOM (Distributed Component Object Model)

- It is a microsoft’s version of the distributed component based computing solutions.
- Popularly know as COM with a long wire.
- Based on the object oriented principle of keeping an object’s interface separate from its implementation.
- Similar to CORBA where COM components interact with each other over the network.
- Every COM component is registered in the Windows registry of the operating system

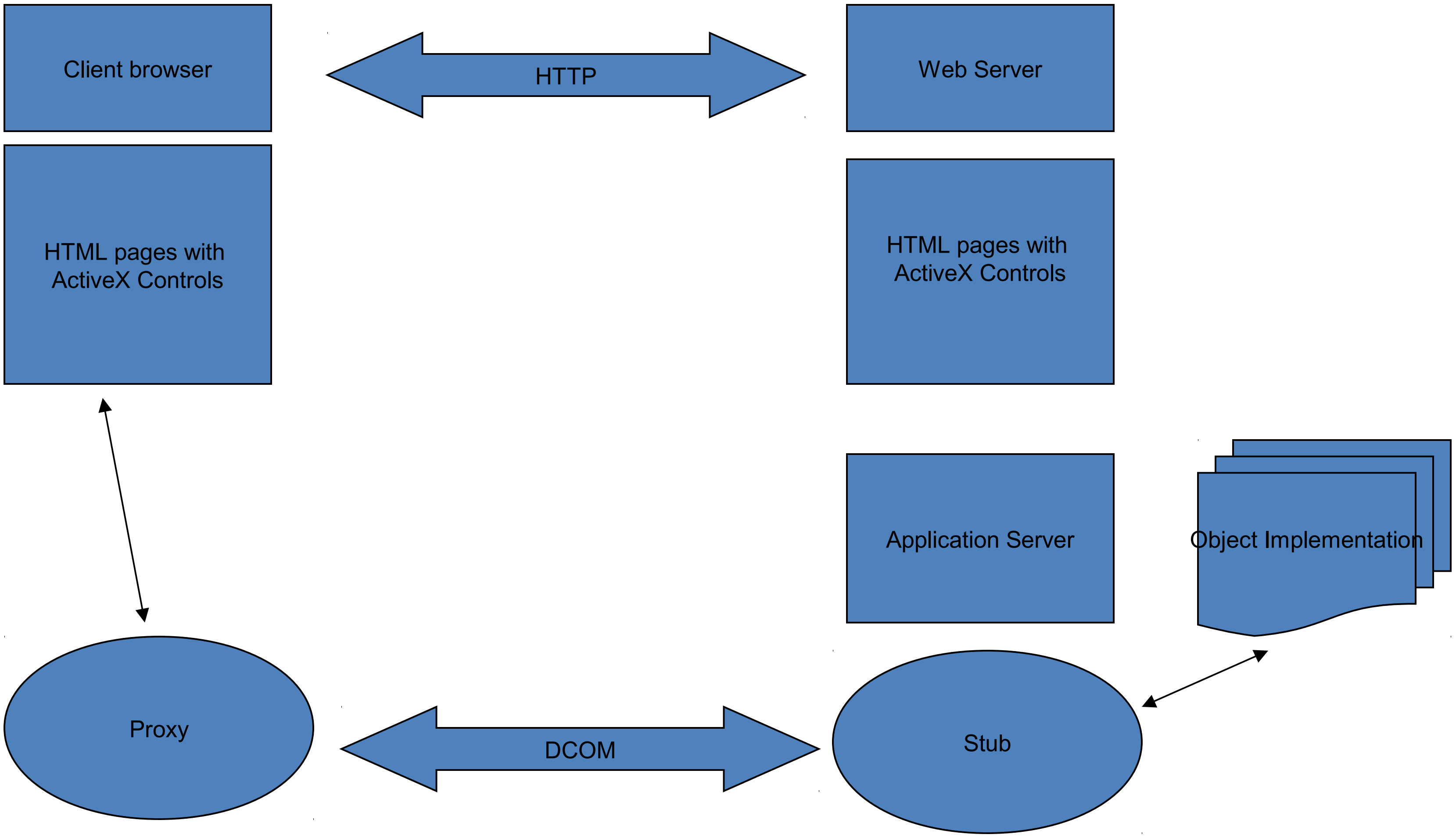
Differences between CORBA and DCOM

- Java applets are the clients in case of CORBA whereas in DCOM ActiveX controls are usually the clients
- Client side infrastructure in DCOM is called **Proxy** and the server side is **Stub.**
- In DCOM when a component wants to invoke a method of another component that is remotely located, the caller has to obtain the **CLSID** of the component to be called.

CLSID is a **class identifier** that identifies any DCOM component all over the world.

CLSID is 128 bits that includes the date and detailed time values when the component was created.

DCOM Architecture



Client browser

HTTP

Web Server

HTML pages with ActiveX Controls

HTML pages with ActiveX Controls

Application Server

Object Implementation

Proxy

DCOM

Stub

Thank You