The Common Object Request Broker Architecture (CORBA)

Based on
Distributed Computing Principles and Applications
by **M.L.Liu**, Pearson Publications, 2004

CORBA

- The Common Object Request Broker Architecture (CORBA) is a standard architecture for a distributed objects system.

- CORBA is designed to allow distributed objects to interoperate in a heterogenous environment, where objects can be implemented in different programming language and/or deployed on different platforms

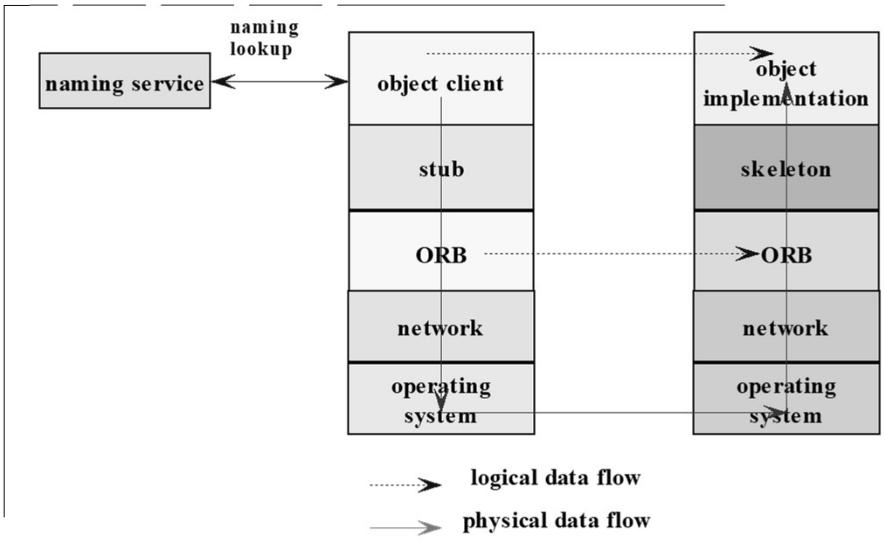
CORBA vs. Java RMI

- CORBA differs from the architecture of Java RMI in one significant aspect:
  - RMI is a proprietary facility developed by Sun MicroSystems, Inc., and supports objects written in the Java programming langugage only.
  - CORBA is an architecture that was developed by the Object Management Group (OMG), an industrial consortium.

CORBA

- CORBA is not in inself a distributed objects facility; instead, it is a set of protocols.
- A distributed object facility which adhere to these protocols is said to be CORBA-compliant, and the distributed objects that the facility support can interoperate with objects supported by other CORBA-compliant facilities.
- CORBA is a very rich set of protocols. We will instead focus on the key concepts of CORBA related to the distributed objects paradigm. We will also study a facility based on CORBA: the Java IDL.

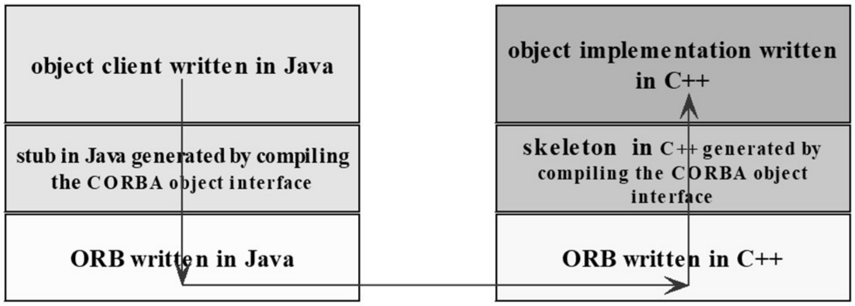
The Basic Architecture



CORBA Object Interface

- A distributed object is defined using a software file similar to the remote interface file in Java RMI.
- Since CORBA is language independent, the interface is defined using a universal language with a distinct syntax, known as the ***CORBA Interface Definition Language*** (***IDL***).
- The syntax of CORBA IDL is similar to Java and C++. However, object defined in a CORBA IDL file can be implemented in a large number of diverse programming languages, including C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLScript.
- For each of these languages, OMG has a standardized mapping from CORBA IDL to the programming language, so that a compiler can be used to process a CORBA interface to generate the proxy files needed to interface with an object implementation or an object client written in any of the CORBA-compatible languages.

Cross-language CORBA application


object client written in Java
stub in Java generated by compiling the CORBA object interface
ORB written in Java
object implementation written in C++
skeleton in C++ generated by compiling the CORBA object interface
ORB written in C++

Inter-ORB Protocols

- To allow ORBs to be interoperable, the OMG specified a protocol known as the ***General Inter-ORB Protocol*** (***GIOP***), a specification which “provides a general framework for protocols to be built on top of specific transport layers.”
- A special case of the protocol is the ***Internet Inter-ORB Protocol*** (***IIOP***), which is the GIOP applied to the TCP/IP transport layer.

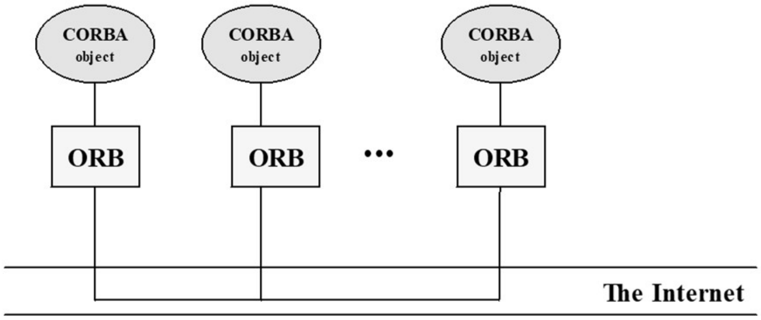
Inter-ORB Protocols

The IIOP specification includes the following elements:

1. **Transport management requirements**: specifies the connection and disconnection requirements, and the roles for the object client and object server in making and unmaking connections.
2. **Definition of common data representation**: a coding scheme for marshalling and unmarshalling data of each IDL data type.
3. **Message formats**: different types of message format are defined. The messages allow clients to send requests to object servers and receive replies. A client uses a Request message to invoke a method declared in a CORBA interface for an object and receives a reply message from the server.

Object Bus

An ORB which adheres to the specifications of the IIOP may interoperate with any other IIOP-compliant ORBs over the Internet. This gives rise to the term “***object bus***”, where the Internet is seen as a bus that interconnects CORBA objects



Object Servers and Object Clients

- As in Java RMI, a CORBA distributed object is exported by an ***object server***, similar to the object server in RMI.
- An ***object client*** retrieves a reference to a distributed object from a naming or directory service, to be described, and invokes the methods of the distributed object.

CORBA Object References

- As in Java RMI, a CORBA distributed object is located using an ***object reference***. Since CORBA is language-independent, a CORBA object reference is an abstract entity mapped to a language-specific object reference by an ORB, in a representation chosen by the developer of the ORB.
- For interoperability, OMG specifies a protocol for the abstract CORBA object reference object, known as the ***Interoperable Object Reference*** (***IOR***) protocol.

Interoperable Object Reference (IOR)

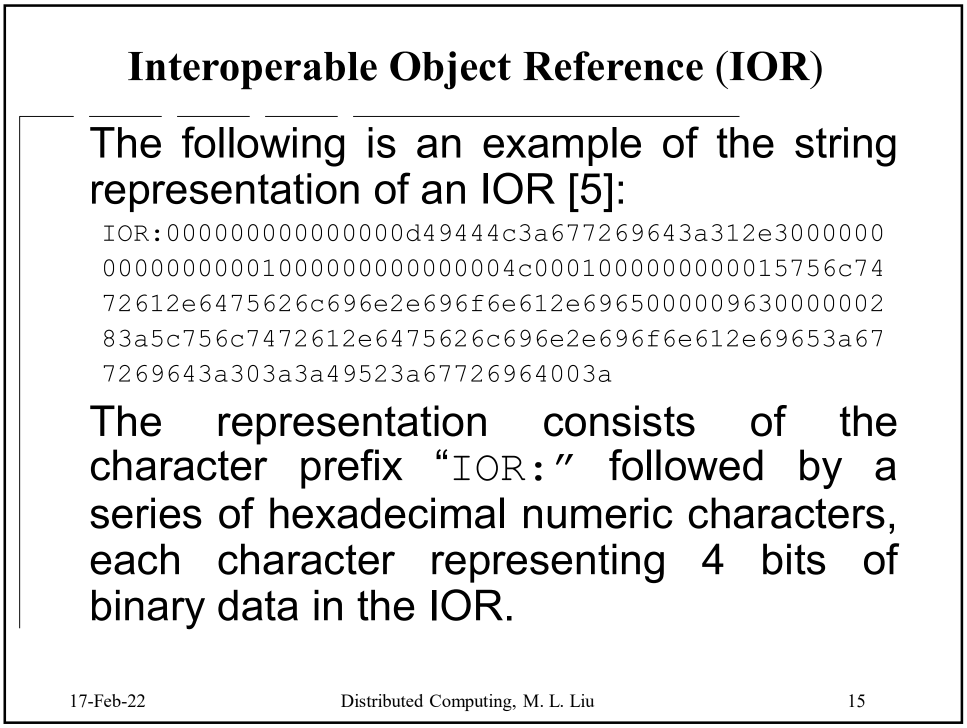
- For interoperability, OMG specifies a protocol for the abstract CORBA object reference object, known as the ***Interoperable Object Reference*** (***IOR***) protocol.
- An ORB compatible with the IOR protocol will allow an object reference to be registered with and retrieved from any IOR-compliant directory service. CORBA object references represented in this protocol are called ***Interoperable Object Reference***s (***IOR***s).

Interoperable Object Reference (IOR)

An IOR is a string that contains encoding for the following information:

- The type of the object.
- The host where the object can be found.
- The port number of the server for that object.
- An object key, a string of bytes identifying the object.

  The object key is used by an object server to locate the object.

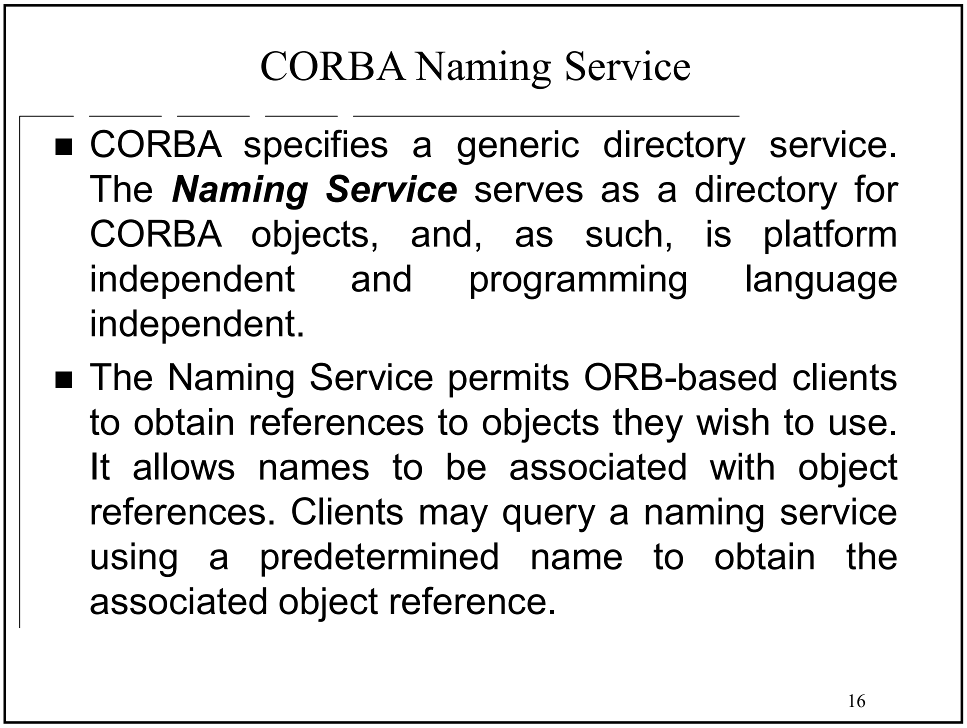


Interoperable Object Reference (IOR)

The following is an example of the string representation of an IOR [5]:

```
IOR:000000000000000d49444c3a677269643a312e3000000
0000000000100000000000000004c00010000000000015756c74
72612e6475626c696e2e696f6e612e696500000963000000002
83a5c756c7472612e6475626c696e2e696f6e612e69653a67
7269643a303a3a49523a67726964003a
```

The representation consists of the character prefix “IOR:” followed by a series of hexadecimal numeric characters, each character representing 4 bits of binary data in the IOR.



CORBA Naming Service

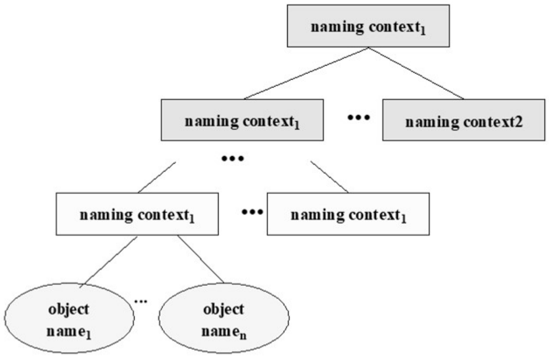
- CORBA specifies a generic directory service. The ***Naming Service*** serves as a directory for CORBA objects, and, as such, is platform independent and programming language independent.
- The Naming Service permits ORB-based clients to obtain references to objects they wish to use. It allows names to be associated with object references. Clients may query a naming service using a predetermined name to obtain the associated object reference.

CORBA Naming Service

- To export a distributed object, a CORBA object server contacts a Naming Service to ***bind*** a symbolic name to the object The Naming Service maintains a database of names and the objects associated with them.
- To obtain a reference to the object, an object client requests the Naming Service to look up the object associated with the name (This is known as ***resolving*** the object name.)
- The API for the Naming Service is specified in interfaces defined in IDL, and includes methods that allow servers to bind names to objects and clients to resolve those names.

CORBA Naming Service

To be as general as possible, the CORBA object naming scheme is necessary complex. Since the name space is universal, a standard naming hierarchy is defined in a manner similar to the naming hierarchy in a file directory



A Naming Context

- A naming context correspond to a folder or directory in a file hierarchy, while object names corresponds to a file.
- The full name of an object, including all the associated naming contexts, is known as a *compound name*. The first component of a compound name gives the name of a naming context, in which the second component is accessed. This process continues until the last component of the compound name has been reached.
- Naming contexts and name bindings are created using methods provided in the Naming Service interface.

A CORBA object name

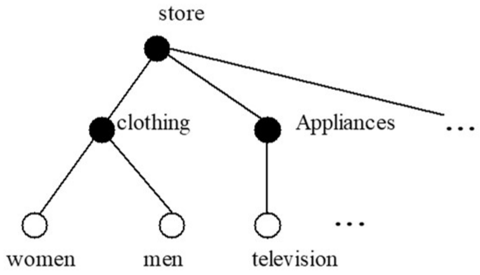
The syntax for an object name is as follows:

```
<naming context > …<naming context><object name>
```

where the sequence of naming contexts leads to the object name.

Example of a naming hierarchy

As shown, an object representing the men’s clothing department is named store.clothing.men, where store and clothing are naming contexts, and men is an object name.



Interoperable Naming Service

The ***Interoperable Naming Service*** (***INS***) is a URL-based naming system based on the CORBA Naming Service.

It allows applications to share a common initial naming context and provide a URL to access a CORBA object.

CORBA Object Services

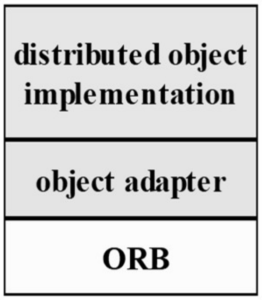
CORBA specify services commonly needed in distributed applications, some of which are:

- ***Naming Service***:
- ***Concurrency Service***:
- ***Event Service***: for event synchronization;
- ***Logging Service***: for event logging;
- ***Scheduling Service***: for event scheduling;
- ***Security Service***: for security management;
- ***Trading Service***: for locating a service by the type (instead of by name);
- ***Time Service***: a service for time-related events;
- ***Notification Service***: for events notification;
- ***Object Transaction Service***: for transactional processing.

Each service is defined in a standard IDL that can be implemented by a developer of the service object, and whose methods can be invoked by a CORBA client.

Object Adapters

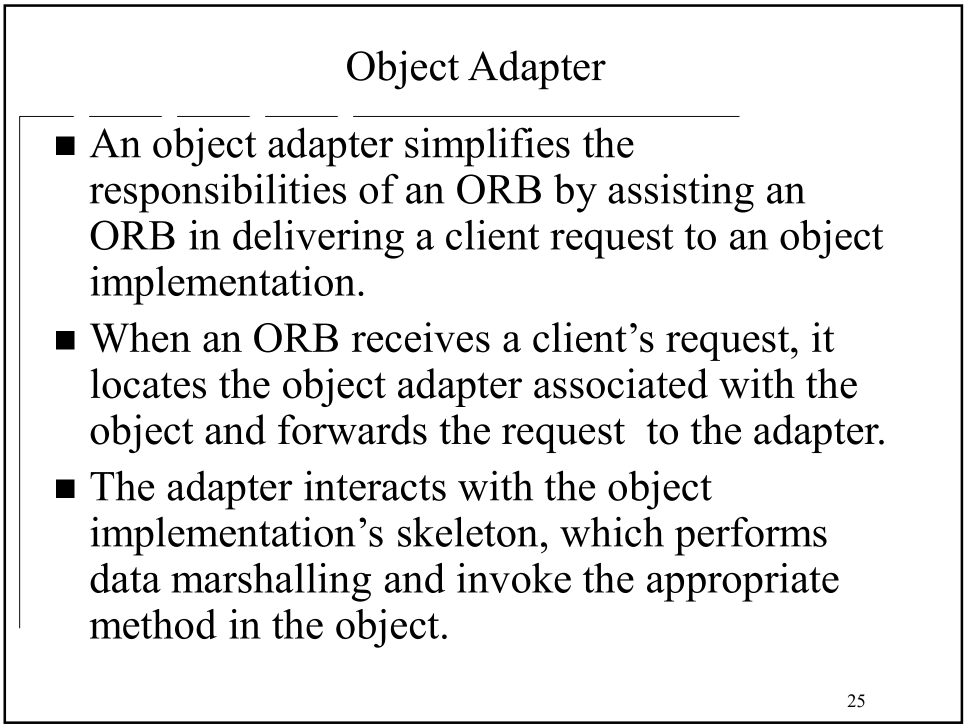
In the basic architecture of CORBA, the implementation of a distributed object interfaces with the skeleton to interact with the stub on the object client side. As the architecture evolved, a software component in addition to the skeleton was needed on the server side: an **object adapter**.



distributed object implementation

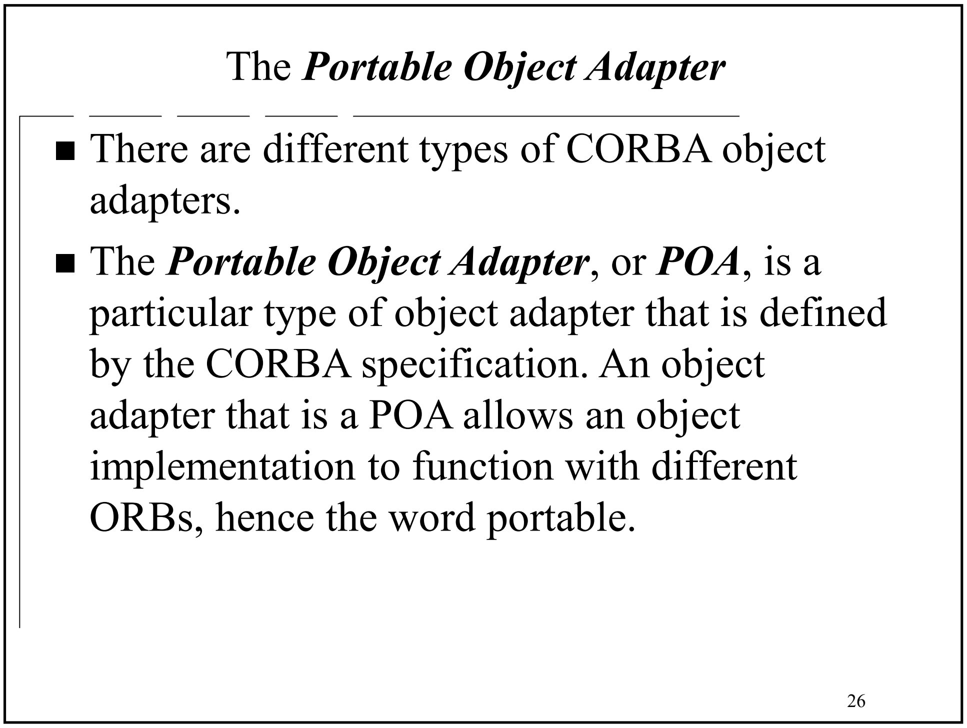
object adapter

ORB



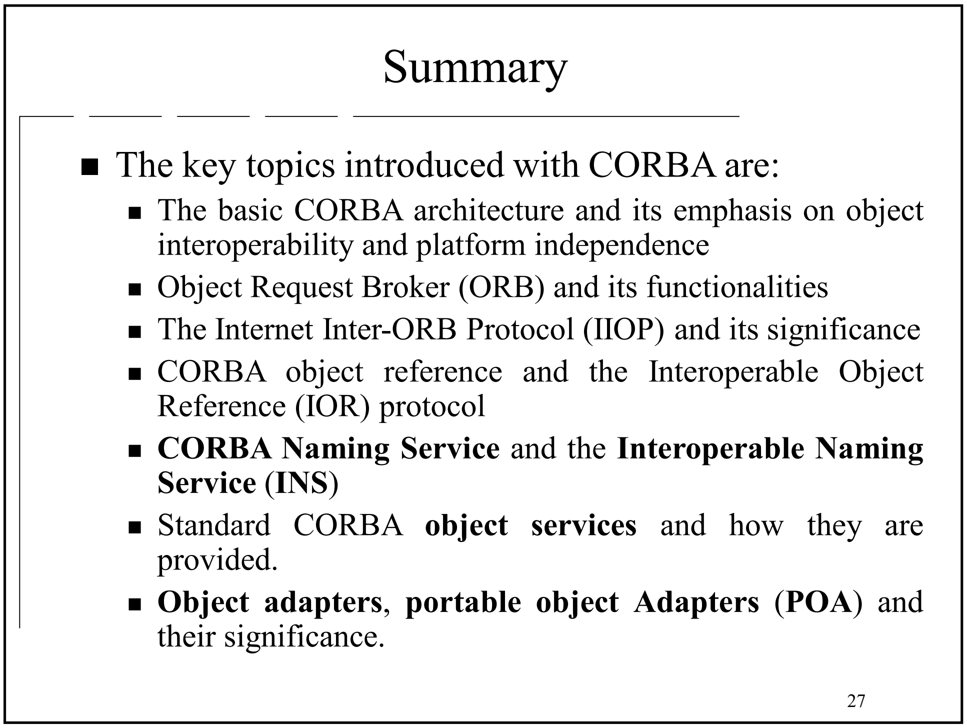
Object Adapter

- An object adapter simplifies the responsibilities of an ORB by assisting an ORB in delivering a client request to an object implementation.
- When an ORB receives a client’s request, it locates the object adapter associated with the object and forwards the request to the adapter.
- The adapter interacts with the object implementation’s skeleton, which performs data marshalling and invoke the appropriate method in the object.



The ***Portable Object Adapter***

- There are different types of CORBA object adapters.
- The ***Portable Object Adapter***, or ***POA***, is a particular type of object adapter that is defined by the CORBA specification. An object adapter that is a POA allows an object implementation to function with different ORBs, hence the word portable.



Summary

- The key topics introduced with CORBA are:
  - The basic CORBA architecture and its emphasis on object interoperability and platform independence
  - Object Request Broker (ORB) and its functionalities
  - The Internet Inter-ORB Protocol (IIOP) and its significance
  - CORBA object reference and the Interoperable Object Reference (IOR) protocol
  - **CORBA Naming Service** and the **Interoperable Naming Service** (**INS**)
  - Standard CORBA **object services** and how they are provided.
  - **Object adapters**, **portable object Adapters** (**POA**) and their significance.