Grid Computing

Paradigms for Distributed Computing 3
DCOM/CORBA
Paul A. Farrell
Fall 2006

The Grid: Core Technologies
Maozhen Li, Mark Baker

From material by Amy Apon at U. Arkansas

DCOM

- Component Object Model (COM) is a binary standard for building Microsoft-based component applications
- DCOM is an extension for distributed client/server applications
- DCOM automatically generates a client-side stub (called proxy in DCOM) and a server-side skeleton (called stub in DCOM)
- Uses Microsoft’s Interface Definition Language (MIDL) interface
- Uses a protocol called the Object Remote Procedure Call (ORPC) to invoke remote COM components.
- The ORPC is layered on top of the OSF DCE RPC specification.
## Advantages/Disadvantages

- DCOM is language independent
- Available on non-Microsoft platforms, but achieved broad popularity only on Windows
- Only supports synchronous communications

## DCOM application steps

- Write an MIDL interface.
- Use an interface compiler (*midl*) to compile the interface to generate a client-side stub and a server-side skeleton.
- Write the COM component to implement the interface.
- Write a DCOM client.
- Compile all the codes.
- Register the COM component with a DCOM server.
- Start the DCOM server.
- Start the DCOM client.

## CORBA

- CORBA is an object-oriented middleware infrastructure from Object Management Group (OMG)
- Automatically generates a client-side stub and a server-side skeleton through an Interface Definition Language (IDL)
- Uses Internet-Inter ORB Protocol (IIOP) to invoke remote CORBA objects
- The Object Request Broker (ORB) is the core of CORBA
  - performs data marshaling and un-marshaling between CORBA clients and objects

![CORBA Diagram](image-url)
Advantages/Disadvantages

- CORBA is independent of location, a particular platform, or programming language
- Supports both synchronous and asynchronous communications
- Has an advanced directory service called COSNaming, which provides the mechanisms to allow the transparent location of objects
- CORBA itself is only an OMG specification
  - Many products available

CORBA steps

- Write a CORBA IDL interface.
- Use an IDL compiler to compile the interface to generate a client-side stub and a server-side skeleton.
- Write a CORBA object to implement the interface.
- Write a CORBA server to register the CORBA object.
- Write a CORBA client.
- Compile all the source codes.
- Start a CORBA name server.
- Start the CORBA server.
- Start the CORBA client.

Summary

Common Features
- An interface is needed for invoking a remote object or a component.
- The complexity of low-level communications is hidden by automatically generating a client-side stub and a server-side skeleton via the interface definition.
- They use proprietary communication protocols to invoke remote objects or components.
- The interface definition is in binary format. It is difficult for client applications to make a query on an interface, such as to find out what kinds of methods are defined, inputs/outputs of each method
- Clients and objects are tightly coupled with their interfaces. For example, changing a part of the client means the other parts, such as the server, need modification.

Shortcomings

- Not based on open standards
- Makes it difficult for them to be ubiquitously taken up in heterogeneous environments

- Ideally, what is needed is an open standards-based middleware infrastructure for building and integrating applications in heterogeneous environments, and Web Services are emerging as such an infrastructure.