CSI3344 Distributed Systems

Workshop Solution 11

Q1. What is the difference between remote objects and distributed objects?

A: A remote object is an object that is hosted by a single server, but whose methods can be invoked by remote clients. In contrast, a distributed object is one whose state may be physically distributed across different servers. An example of distributed objects is that provided by Globe. Most object-based systems, however, support only remote objects.

Q2. Why is it useful to define the interfaces of an object in an Interface Definition Language?

A: There are several reasons. First, from a software-engineering point of view, having precise and unambiguous interface definitions is important for understanding and maintaining objects. Secondly, IDL-based definitions come in handy for generating stubs. And thirdly, if an IDL definition has been parsed and stored, supporting dynamic invocations becomes easier, as the client-side proxy can be automatically constructed at runtime from an interface definition.

Q3. Some implementations of distributed-object middleware systems are entirely based on dynamic method invocations. Even static invocations are compiled to dynamic ones. What is the benefit of this approach?

A: Realizing that an implementation of dynamic invocations can handle *all* invocations, static ones become just a special case. The advantage is that only a single mechanism needs to be implemented. A possible disadvantage is that performance is not always as optimal as it could be had we analyzed the static invocation.

Q4. What is ORB in CORBA? What is the functionality of ORB?

A: (See slides No. 5 to 9 in Lecture 11).

END OF THE WORKSHOP SOLUTION