ActiveX (DCOM) VS RMI

Aseem Sharma
Ricardo Pravia
Carnegie Mellon University
School of Computer Science

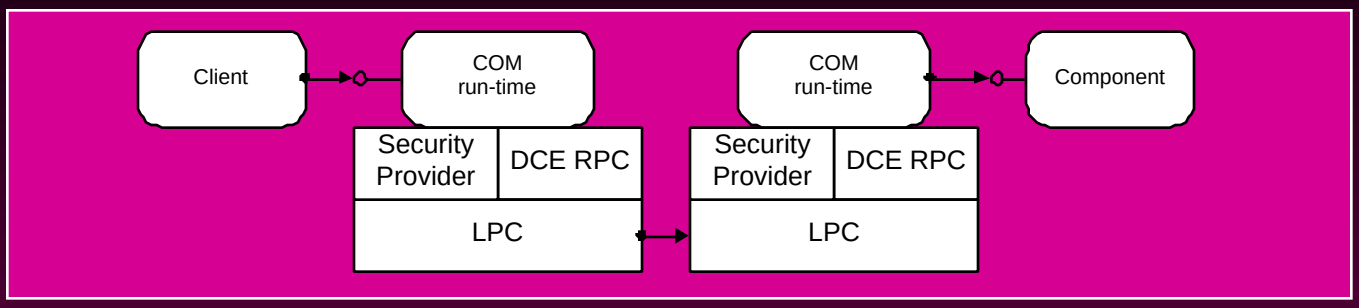
ActiveX (DCOM) Introduction

- ActiveX is a set of technologies that enables software components to interact with one another in a networked environment, regardless of the language in which they were created. ActiveX is built on the Distributed Component Object Model (DCOM) as explained on the next page .
- developers can create components in any programming language, integrate them with any scripting language, and run those components from many types of applications, including Web browsers as well as many of the world's most popular business applications

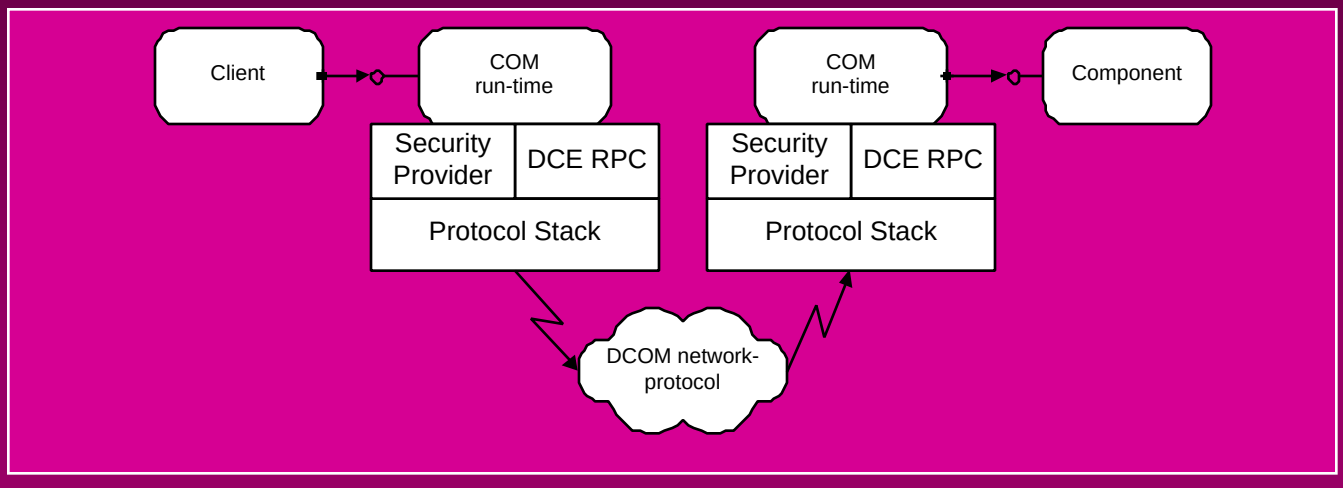
Component Object Model(COM)

- DCOM is an extension of the Component Object Model (COM). COM defines how components and their clients interact. This interaction is defined such that the client and the component can connect without the need of any intermediary system component. The client calls methods in the component without any overhead whatsoever
- In today's operating systems, processes are shielded from each other. A client that needs to communicate with a component in another process cannot call the component directly, but has to use some form of inter-process communication provided by the operating system. COM provides this communication in a completely transparent fashion: it intercepts calls from the client and forwards them to the component in another process

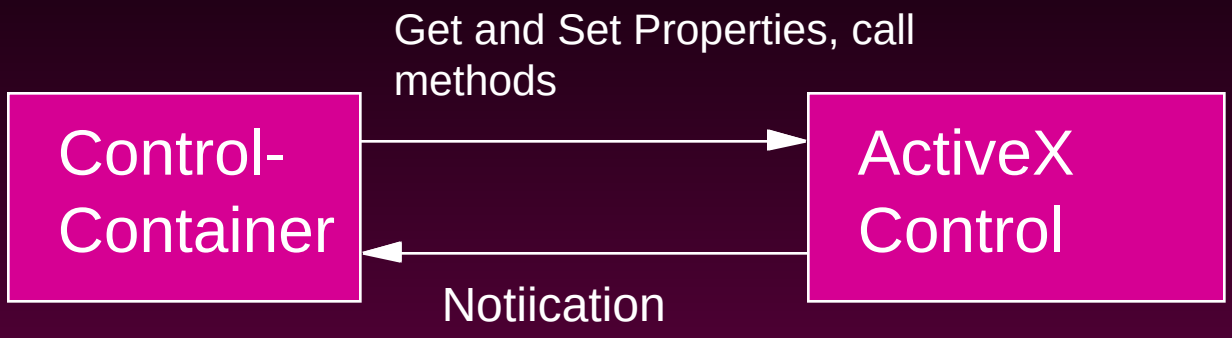
COM architecture



- When client and component reside on different machines, DCOM simply replaces the local Inter-process communication with a network protocol. Neither the client nor the component are aware that the wire that connects them has just become a little longer.

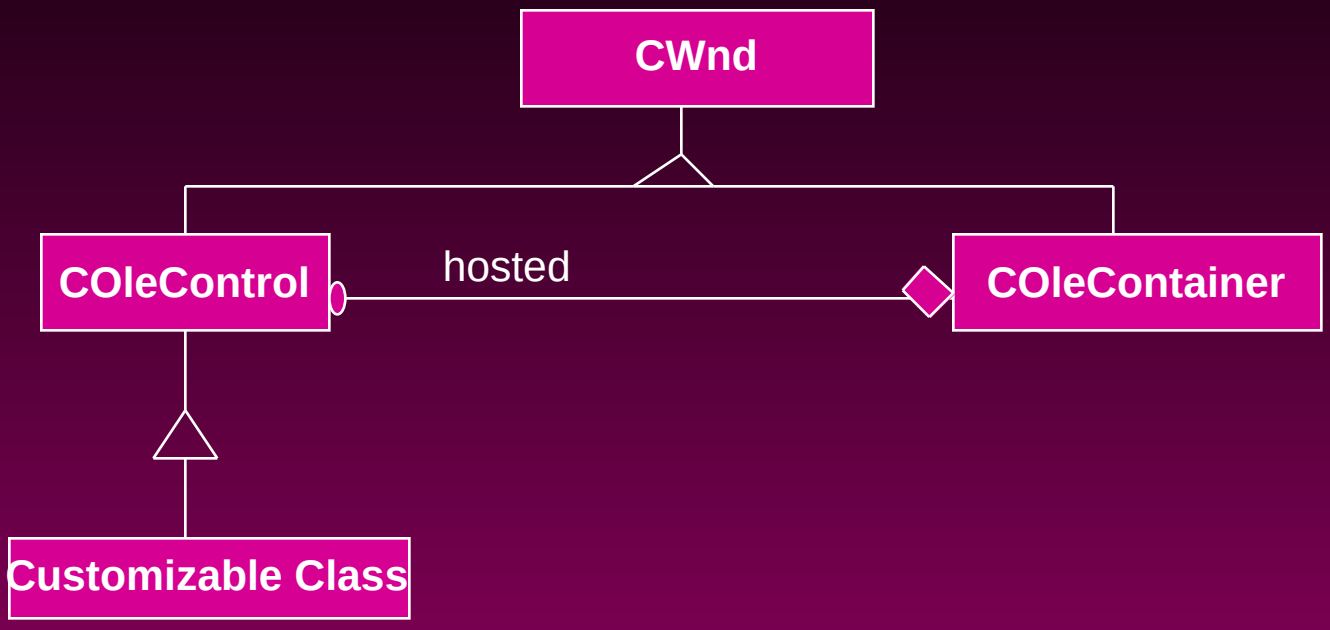


The Framework for ActiveX Controls



- An ActiveX control is contained inside a container much like an embeded and linked object.
- In case of ActiveX, the most common Control-Container is the Internet Explorer 3.0
- The Control publishes a set of public member functions (services) that are called by the container.
- The Control can initiate conversation to the container only through notification.

The Object Model for Control and Container



The Cool thing about ActiveX is that an object from COleControl could be a remote object.

Industry Support

- Component availability. There are lots of components today that can be used using ActiveX.
- Development tools. ActiveX components can be created or used with a myriad of development tools spanning a few different programming languages.
- Component containers. ActiveX components can be "hosted" in many of the most popular applications used every day by tens of millions of people, including Lotus Notes®, Lotus SmartSuiteTM, Microsoft Office and Microsoft Internet Explorer 3.0

Other Advantages

- The availability of off-the-shelf components helps speed delivery time.
- Cross-platform development. In conjunction with Macromedia Inc. and MetroWerks Inc., Microsoft is working to bring ActiveX to the Macintosh®, and with Bristol Technology Inc., Mainsoft Corp. and Software AG to bring ActiveX to UNIX®.

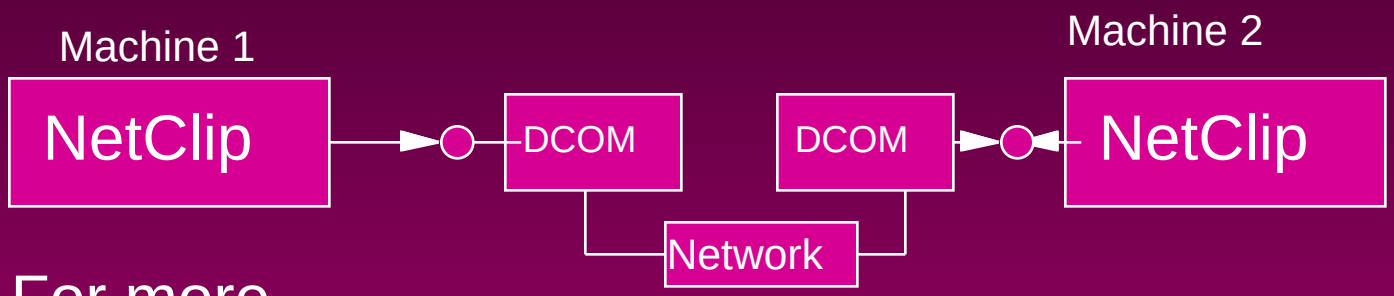
One of Many examples

- Play audio in real-time over modems -- no downloads -- using the RealAudio Player. The RealAudio Control for ActiveX allows you to incorporate RealAudio controls directly into your Web pages or VisualBasic applications.

```
SCRIPT language="VBS">
ub PlayPause_OnClick()
        RAOCX.DoPlayPause
nd Sub
ub Stop_OnClick()
        RAOCX.DoStop
nd Sub
ub RAOCX_OnShowStatus(byVal newText)
        StatusText.value = newText
nd Sub
```

Example (Cntd.)

- For the complete list of attributes that can be changed and methods that can be performed on the ActiveX control, refer to http://www.realaudio.com/help/sdk/sdkactivex/index.htm
- Example of DCOM: A Netclip application object shares a clipboard with another NetClip application on another machine.



For more

Java RMI

- Java’s solution to distributed computing.
- RMI enables method invocation for objects on different VM’s, possibly on different hosts.

How it works

1) Create an Interface (a class with only methods and no implementation code)
2) Declare it Remote
3) Create a class that implements the interface.
4) Export it to some Registry with some particular name
5) Clients can then lookup in that Registry (itself a remote object) for the name and receive a remote object (a reference to a remote object)

Create a remote interface(steps 1 & 2)

- Use Java's own interface class:

```
public interface myServer extends Remote {
    boolean ping();
    String parseString(String s);
}
```

- Uses Java's own language to achieve its purpose
- No tools necessary

Implement and Register (steps 3&4)

- Create a class that implements the interface:

```
public class myServerImpl extends UnicastRemoteObject
                          implements myServer{
  ...<implement methods>...
  public static void main(String args[]) {
    System.setSecurityManager(new RMISecurityManager);
    myServer server = new MyServer();
    Naming.rebind(“//myhost/CoolServer”, obj);
  }
}
```

- URL type of resource location for both client and server.
- Note that there is only one Security Manager per VM (not per remote object)

Client Access to Remote Object (step 5)

- Simple Client connection:

```
... <client code> ...
myServer _server;
_server=(myServer)Naming.lookup(“//remoteHost/CoolServer”);
```

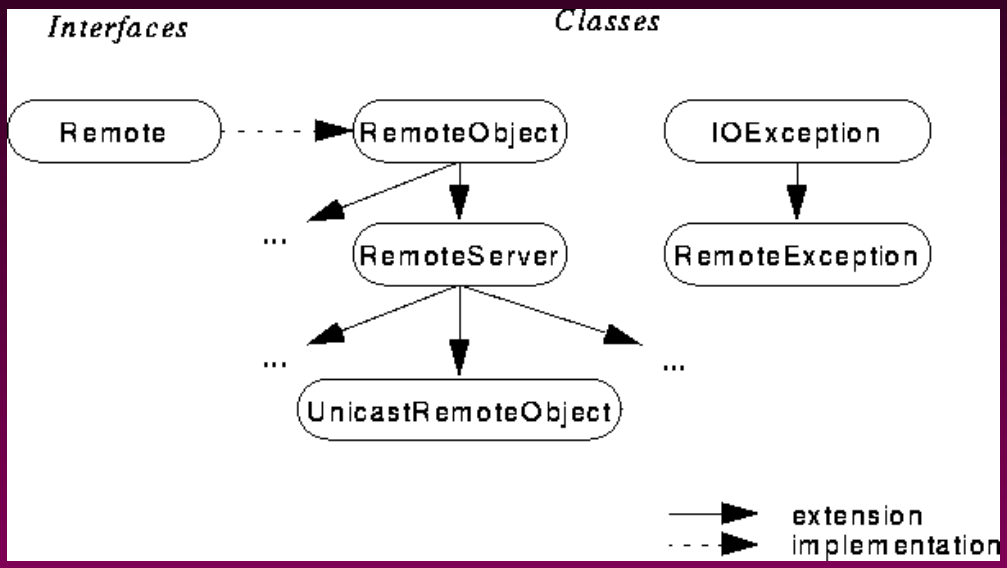
- Simple URL locator
- Also methods available for Applets that use RMI

Java RMI

- Extends very nicely from the Java language
- In consequence, great features come with RMI such as type checking, GC, etc.
- Robust and extensible security
- Well defined and extensible architecture

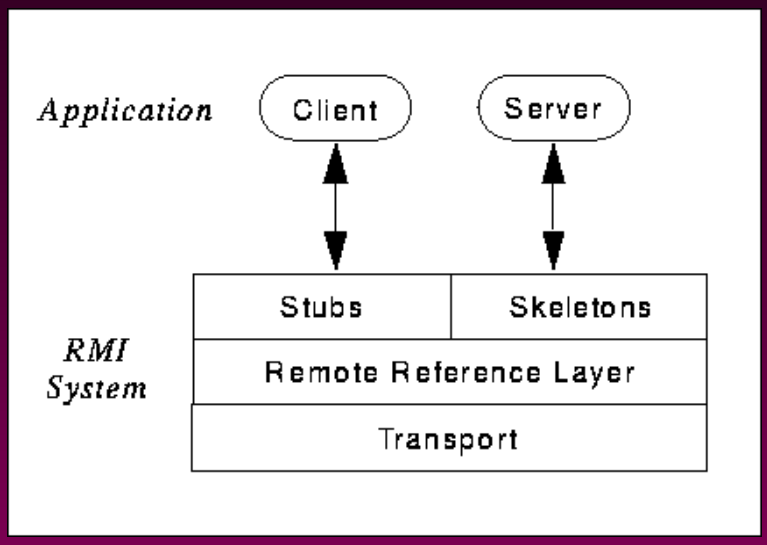
Simple but powerful

- Extensible architecture



Simple but powerful (cont)

- Layered Architecture



- Allows failure transparency, transparent fallback procedures.

DCOM vs RMI

- Language Independence
- Lots of industry support already
- The framework is inherently component oriented.
- Hence, development cycle is small.
- Lots of development tools.
- Lots of emphasis on graphical controls.
- Incredible flexibility

- Platform Independence (JAVA)
- All Java types available
- Easy handing of remote objects
- Very robust class loading mechanism (clients may discover a new remote object at runtime)
- Extensible Security
- Fast learning curve (Simple but powerful)
- Object Serialization