Modulo II – Socket, RMI e Corba

Prof. Ismael H F Santos

Ementa

- Sistemas Distribuídos
  - *Cliente-Servidor*

SCD – CO023

Client-Server

SCD – CO023

Sistemas Distribuídos

Middleware layers

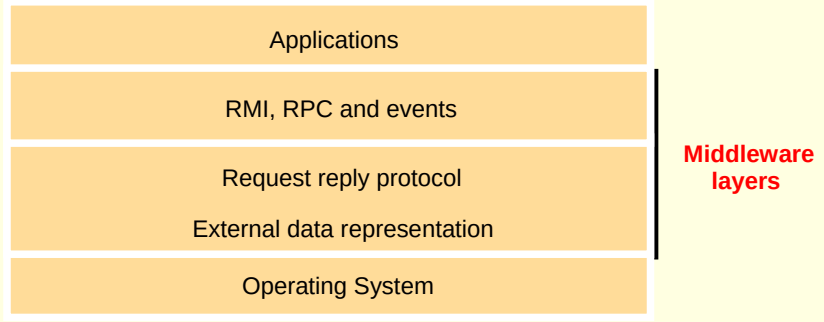
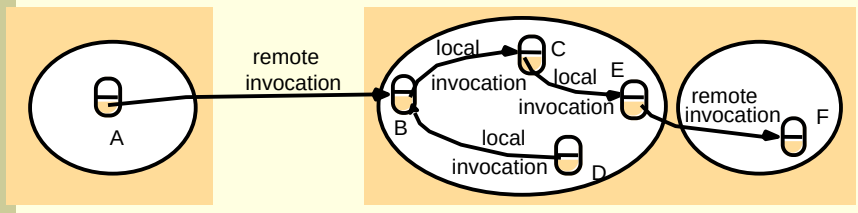


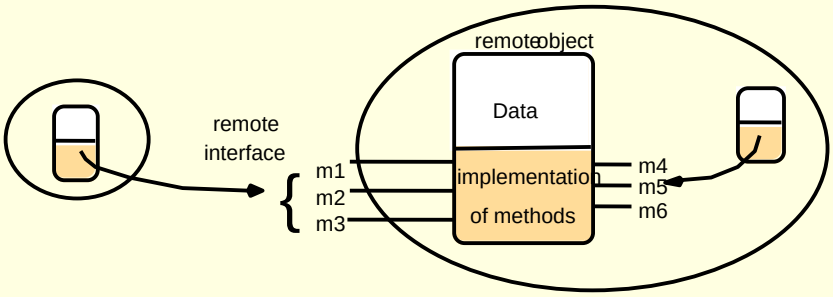
Figure 5.2
CORBA IDL example

```
// In file Person.idl
struct Person {
        string name;
        string place;
        long year;
} ;
interface PersonList {
        readonly attribute string listname;
        void addPerson(in Person p) ;
        void getPerson(in string name, out Person p);
        long number();
};
```

Remote and local method invocations



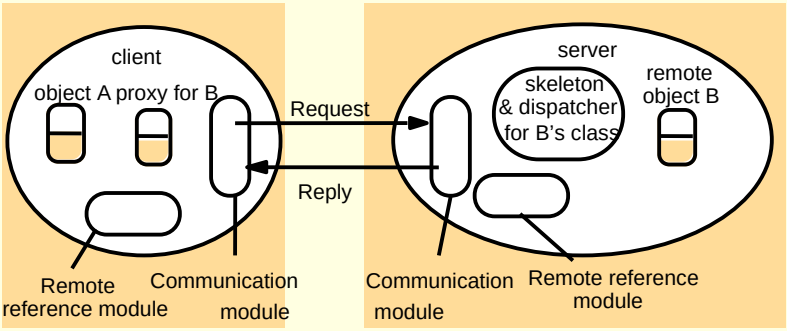
A remote object and its remote interface



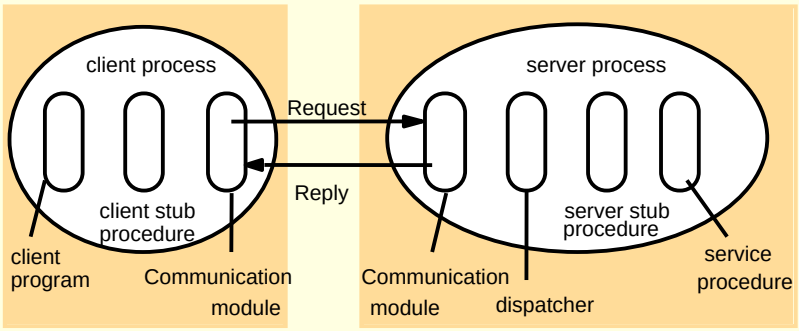
Invocation semantics

Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

The role of proxy and skeleton in remote method invocation



Role of client and server stub procedures in RPC



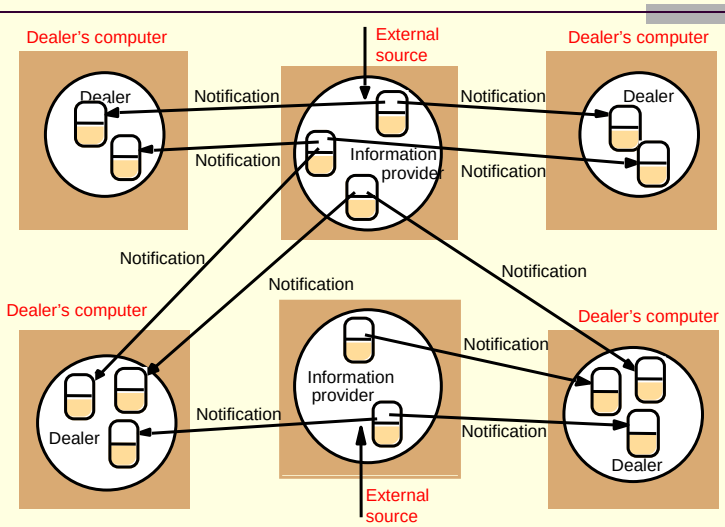
Files interface in Sun XDR

```
const MAX = 1000;
typedef int FileIdentifier;
typedef int FilePointer;
typedef int Length;
struct Data {
    int length;
    char buffer[MAX];
};
struct writeargs {
    FileIdentifier f;
    FilePointer position;
    Data data;
};
```

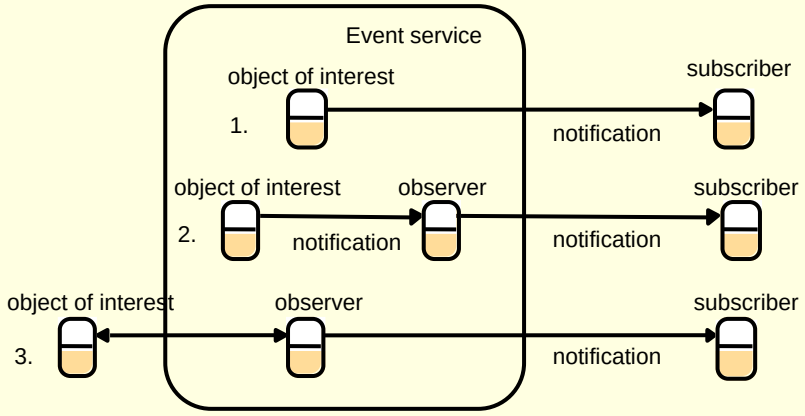
```
struct readargs {
    FileIdentifier f;
    FilePointer position;
    Length length;
};

program FILEREADWRITE {
  version VERSION {
    void WRITE(writeargs)=1;   1
    Data READ(readargs)=2;     2
  }=2;
} = 9999;
```

Dealing room system



Architecture for distributed event notification



Java Remote interfaces *Shape* and *ShapeList*

```
import java.rmi.*;
import java.util.Vector;
public interface Shape extends Remote {
    int getVersion() throws RemoteException;
    GraphicalObject  getAllState() throws RemoteException;      1
}
public interface ShapeList extends Remote {
    Shape newShape(GraphicalObject g) throws RemoteException;   2
    Vector allShapes() throws RemoteException;
    int getVersion() throws RemoteException;
}
```


The *Naming* class of Java RMIregistry

void rebind (String name, Remote obj)
This method is used by a server to register the identifier of a remote object by name, as shown in Figure 15.13, line 3.

void bind (String name, Remote obj)
This method can alternatively be used by a server to register a remote object by name, but if the name is already bound to a remote object reference an exception is thrown.

void unbind (String name, Remote obj)
This method removes a binding.

Remote lookup(String name)
This method is used by clients to look up a remote object by name, as shown in Figure 15.15 line 1. A remote object reference is returned.

String [] list()
This method returns an array of Strings containing the names bound in the registry.

Java class *ShapeListServer* with *main* method

```
import java.rmi.*;
public class ShapeListServer{
    public static void main(String args[]){
        System.setSecurityManager(new RMISecurityManager());
         try{
            ShapeList aShapeList = new ShapeListServant();                1
                Naming.rebind("Shape List", aShapeList );                 2
            System.out.println("ShapeList server ready");
            }catch(Exception e) {
            System.out.println("ShapeList server main " + e.getMessage());}
    }
}
```


Java class *ShapeListServant* implements interface *ShapeList*

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.Vector;
public class ShapeListServant extends UnicastRemoteObject implements ShapeList
{
     private Vector theList;          // contains the list of Shapes          1
     private int version;
    public ShapeListServant()throws RemoteException{...}
    public Shape newShape(GraphicalObject g) throws RemoteException {      2
        version++;
        Shape s = new ShapeServant( g, version);                           3
        theList.addElement(s);
        return s;
    }
    public  Vector allShapes()throws RemoteException{...}
    public int getVersion() throws RemoteException { ... }
}
```

Java client of *ShapeList*

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Vector;
public class ShapeListClient{
   public static void main(String args[]){
     System.setSecurityManager(new RMISecurityManager());
     ShapeList aShapeList = null;
     try{
          aShapeList  = (ShapeList) Naming.lookup("//bruno.ShapeList") ;   1
          Vector sList = aShapeList.allShapes();                           2
     } catch(RemoteException e) {
       System.out.println(e.getMessage());
     } catch(Exception e) {
       System.out.println("Client: " + e.getMessage());}
   }
}
```


Classes supporting Java RMI