Distributed Programming

- Low level: sending data among distributed computations
  - Network is visible to the programmer
  - Programmer must deal with many details

- Higher level: supporting invocations among distributed computations
  - Network is invisible to the programmer
  - Programmer focuses on application

Distributed Data Communication

Java Classes

- Socket
  - DataOutputStream (PrintStream)

Basic Socket Usage

**Client**

- Establish Socket Connection
  - `Socket cs;`
  - `int portno = 5678;`
  - `cs = new Socket(`server' , portno);`

- Establish Data Streams
  - `clin = new DataInputStream(cs.getInputStream());`
  - `clout = new PrintStream(cs.getOutputStream());`

**Server**

- Establish Socket Connection
  - `ServerSocket ss;`
  - `Socket sin;`
  - `ss = new ServerSocket(portno);`
  - `Socket sin = s.accept();`

- Establish Data Streams
  - `srout = new PrintStream(sin.getOutputStream());`
  - `srin = new DataInputStream(sin.getInputStream());`

Client Side Code

```java
class SocketTest {
    public static void main(String[] args) { try {
        Socket t = new Socket("java.sun.com", 13);
        DataInputStream is = new DataInputStream(t.getInputStream());
        boolean more = true;
        while (more) {
            String str = is.readLine();
            if (str == null) more = false;
            else
                System.out.println(str);
        }
    } catch (IOException e) { System.out.println("Error" + e); } }
}
```

Server Side Code

```java
class EchoServer {
    public static void main(String[] args) { try {
        ServerSocket s = new ServerSocket(8189);
        Socket s1 = new Socket("java.sun.com", 13);
        DataInputStream is = new DataInputStream(s1.getInputStream());
        PrintStream out = new PrintStream(s.getOutputStream());
        System.out.println("Hello! Enter BYE to exit.");
        boolean done = false;
        while (!done) {
            String str = is.readLine();
            if (str == null) done = true;
            else {
                out.println("Echo: " + str + "\r\n");
                if (str.trim().equals("BYE"))
                    done = true;
            }
        }
        incoming.close();
    } catch (Exception e) { System.out.println(e); }
}
```
Remote Procedure Call

### Remote Procedure Call Issues
- generating stubs
- serialization or arguments and return values
- heterogeneity of data representations
- locating servers in a distributed environment
- authentication of called and calling procedures
- semantics of invocation (at-most-once, at-least-once)

Serialization

- How to represent base types (i.e., int)
- How to represent structured types (arrays)
- How to deal with references (pointers)
- How to treat duplicated objects

Simple IDL Example

```idl
class Counter
{
    interface Count
    {
        attribute long sum;
        long increment();
    }
};
```

IDL Elements

```idl
class Counter
{
    attribute long sum;
    long increment();
};
```
Remote Object Systems

invoking object

proxy objects

invoked object

network objects

Corba

Goal: interoperability among application components
*written in different programming languages
*executing on heterogeneous architectures
*communicating over different networks

Corba: Common Object Request Broker Architecture
ORB: Object Request Broker

Role of the Object Request Broker

Application interfaces: interfaces for a specific application
Domain interfaces: interfaces shared across applications in a given application domain (publishing)
Common Facilities: generic services that might be needed in several domains (document structure)
Object Services: commonly needed across all applications (e.g., naming, trading)

From Doug Schmidt

Elements of Corba

From Kate Keahey (kksiazek@cs.indiana.edu)

Role of IDL in Corba

From Object Management Group

Elements of Corba

From Doug Schmidt
Corba and Java

Corba is still needed to fill in the gaps between Java and system developed in other languages.