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

- DataInputStream
- DataOutputStream
- PrintStream
- Socket

Basic Socket Usage

Client

```java
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

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

Calling Procedure

args

results

Called Procedure

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

Issues:
- 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

transforming a typed, highly structured object into a stream of bytes.

IDL Elements

module Counter
{
    interface Count
    {
        attribute long sum;
        long increment();
    };
};

From: Ole Arthur Bernsen
Remote Object Systems

proxy objects

invoking object

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
Domain Interfaces
Common Facilities

Object Request Broker

Object Services

- 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)

Elements of Corba

Role of IDL in Corba

IDL: Interface Definition Language

IDL Repository

Object Interface

Implementation Interfaces

Object Interface

IDL Repository

Elements of Corba

From Object Management Group
Corba and Java

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