Distributed Systems

The two critical differences between centralized and distributed systems are:

- absence of shared memory
- absence of a global clock

We will study:

- how programming mechanisms change as a result of these differences
- algorithms that operate in the absence of a global clock
- algorithms that create a sense of a shared, global time
- algorithms for creating shared memory semantics in distributed systems

Distributed Data Communication

Basic Socket Usage

**client**

```java
// Establish Socket Connection
Socket cs;
int portno = 5678;
cs = new Socket("server", portno);

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

**server**

```java
// Establish Socket Connection
ServerSocket ss;
int portno = 5678;
ss = new ServerSocket(portno);

// Establish Data Streams
PrintStream sinOut = new PrintStream(ss.getInputStream());
PrintStream sinIn = new PrintStream(sinOut.getOutputStream());
```

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 = incoming = s.accept();
            DataInputStream in =
                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
                    System.out.println("Echo: " + str + "\r\n");
                if (str.trim().equals("BYE") )
                    done = true;
            }
            incoming.close();
        }
        catch (IOException 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)

Interface Definition Language

Remote Object Systems

Corba