Distributed Systems

- Course Outline
  - Introduction
  - RMI and Web Services
  - J2EE and EJB Technology
  - Java Messaging Service
  - Node.js (Server-side JS)
  - Virtualisation
  - Cloud Computing
  - Load Balancing
  - Lab Work & Assignments

Introduction

» Client / Server Architectures

- A two-tier architecture is where a client talks directly to a server, with no intervening server.
- It is typically used in small environments (less than 50 users).
  - A common development error is to prototype an application in a small, two-tier environment, and then scale up by simply adding more users to the server.
  - This approach will usually result in an ineffective system, as the server becomes overwhelmed.
  - To properly scale to hundreds or thousands of users, it is usually necessary to move to a three-tier architecture.

Introduction

– Client and Server using TCP/IP protocols to communicate:

– Information can flow in either or both directions.

– The Client and Server each interact with a transport layer protocol.

Introduction

» Three-Tier Architecture

– A three-tier architecture introduces a server (or an "agent") between the client and the server.

– The role of the agent is many fold.

- It can provide translation services (as in adapting a legacy application on a mainframe to a client/server environment).
- Metering services (as in acting as a transaction monitor to limit the number of simultaneous requests to a given server).
- Intelligent agent services (as in mapping a request to a number of different servers, collating the results, and returning a single response to the client).

Introduction

» Network Programming Paradigms

- Practically all network programming is based on a Client / Server model.
- The only real difference in paradigms is the *level* at which the programmer operates.
- Sockets API provides direct access to the available transport layer protocols.
- RPC is a higher level abstraction that hides some of the lower level complexities.
- Other approaches are possible…

Introduction

» Advantages of each approach

- Sockets are probably the best known and most widely used paradigm. However, problems of data incompatibility across platforms can arise.
- RPC libraries aim to solve some of the basic problems with sockets and provide a level of transport independence.
- Neither approach works very well with modern applications (Java RMI and other modern technologies e.g. web services are better).