SOEN 423

Lecture Notes

## Lecture 1: Ch.1 & 2

* Brush up on multi-threaded programming.
* Individual assignments (15%)
* Team project (15%)
* Mid-Term (20%)
* Final Exam (50%)
* Distributed systems have n independent hosts.
* A distributed application is supposed to run on multiple hosts.
* Components can be a process, module, package, etc.
* Components run independently and work together through means of communication via a network.
* Hosts run software components on its respective hardware via middleware, which runs on top of the host’s network operating system.
* A networked computer system is not necessarily a distributed computer system.
* A distributed system may run on multiple networks.
* The main difference between a networked system and distributed system is the fact that each component recognize each other by name, NOT location, or software language, etc.
* Sitting on top of the network operating system, is the middleware.
* Everything is transparent.

## Definition:

A distributed system is a collection of autonomous **hosts** that are connected through a computer **network**. Each host executes **components** and operates a distribution **middleware**, which enables the components to coordinate their activities in such a way that ***users perceive the system as a single, integrated computer facility***.

* SPEED is not the deciding factor for designing an application to be distributed.
* So then, what is?
* It can run anywhere.
* Hardware is scalable (i.e. memory, storage, compute power, etc.).
* Scalable systems react to the load of the system by creating and running more and more components as the load increases, and killing them as the load decreases, via the middleware.

## Requirements that are common for distributed systems:

* Concurrency
* Resource Sharing
* Openness \*
* Scalability
* Fault Tolerance
* Transparency \*

## Lecture 2: Ch.1 & 2

* Distributed computing paradigms:
  + Client-Server
  + Peer-to-Peer
  + Distributed Agents
  + Distributed Object Spaces
  + etc.
* This course focuses on the client-server paradigm ***only***.

## Lecture 3: Ch.3 & 4

* Remote Procedure Call (i.e. RPC) is fundamental.
  + Remote Method Invocation (i.e. RMI): Object-Oriented RPC.
  + Network Services: RPC among mostly homogeneous processes (i.e. over LANs)
  + CORBA: RPC among heterogeneous processes (i.e. over WANs using IIOP (i.e. Internet Inter-ORB Protocol (i.e. CORBA) or Internet Inter-Operable Protocol) ).
  + Web Services: RPC over the web using HTTP.
* Server Design: Server characteristics that are achieved through process replication;
  + Scalable: Automatically create server processes to distribute client requests.
  + Fault Tolerant: Maintain multiple server processes so that a faulty can be taken over by a good one.
  + Highly Available: Maintain multiple server processes so that a client request can be handled without delay.

## Lecture 4: Ch.3 & 4