**Distributed Systems Revision Week 1**

# What are distributed Systems?

Distributed Systems are multiple computers working together on the same task(s), connected by network that allows them to exchange information and coordinate information through message passing. The latter allows the transparency of independent failures and lack of global clocks. The key reason for distributed systems is the shared resources allow with such design, such as printers.

# Characteristics

Characteristics of distributed systems are: Concurrency, independent failure of components gracefully, synchronisation of local systems and heterogeneity (variations in component systems), openness, security, scalability and transparency.

# Examples of Distributed Systems

* Intranet, WWW, Internet
  + Using the satellite or wired circuits as the transferable medium using HTTP protocol.
  + Uses a backbone (high transmission capacity, such as fibre optics)
  + Uses of LAN and interconnected systems
* Online Multiplayer games
  + Real time
  + Control fields and information must be consistent across all users
* The Web
  + Transparency across the resources
  + Scalable services

# DS Challenges

Synchronisation across the nodes of the system is vital and very difficult as it is impossible for the nodes to know the status of all the other nodes. Knowledge across the systems is never synchronised completely in real time in any multitude of systems. Being cost effective is difficult and finally, it is impractical to ensure all nodes are up to date with the latest information.

Problems with heterogeneity, incorporating variety and differences, includes networks (HTTP), hardware (big endian), operating systems (windows), programming languages (different representations for storage), developers (agreed standards), middleware, Mobile code etc.

Openness is the concept that systems can be developed the same in several different ways, combined with maintainability. It conforms to standards agreed by the community.

Security has three issues – Confidentially, Integrity and Availability against denial of services, trojans, spoofing etc.

Scalability – Control cost of physical costs and performance, software resources management, avoiding bottle neck

Failure Handling – Detecting, Masking, Tolerating, Recovery, Redundancy, availability, Concurrency etc.

# Transparency

* Access – enable local / remote resources
* Location
* Concurrency
* Replication
* Failure
* Mobility
* Performance
* Scaling

# Issues around Distributed Systems

* Remote Method Invocation
* Load Balancing
* Transparent Failover
* Backend intergradation
* Transactions
* Clustering
* Dynamic Redeployment
* Clean Shutdown
* Logging and Auditing
* System management
* Threading
* Object Lifecycle
* Resource Pooling
* Caching