Distributed System Models

- **Architectural models**
  - Concerned with placement of parts and their relationships
  - Defines how these parts map down onto the network and computers.

- **Fundamental models** formalize properties of the systems (e.g.: correctness, reliability, etc…)

- **Distributed system characteristics addressed by:**
  - Interaction model
  - Failure model
  - Security model
Difficulties For / Threats To Dist. Sys.

- Widely varying models of use
  - Workload has wide variation
  - Poor connectivity of some parts of system
  - Applications have different requirements
    - Bandwidth and latency
- Wide range of system environments
  - Heterogeneous hardware, OS, networks
  - Varying network performance
  - Widely differing system scales
- Internal problems – clocks, data, component failure
- External problems – attacks, data integrity, secrecy
Lamport’s Definition of a DS

- Lamport once defined a distributed system as:
  - “One on which I cannot get any work done because some system I never heard of has crashed.”

- Applications need to adapt gracefully in the face of partial failure.

- An example of a distributed system technology that will lead to Lamport’s issue is NFS. How many of us have ever seen a set of workstations freeze because the NFS server failed?
  - Distributed file systems are hard, especially with respect to adaptation to failure.
Architectural Models

- Ensure that the **structure** meets requirements.
- Simplify and abstract functions of individual components of a distributed system. Then consider:
  - How these are placed amongst a set of networked computers. We seek to define useful patterns to drive data distribution, workload distribution.
  - Inter-relationships between components, their functional roles and communication patterns.
- **Classification aids in simplification.**
  - Servers, clients, peers.
  - Classification identifies responsibilities, behavior, workload and failure properties.
  - Analysis is used to specify placement based to meet objectives.
System Architectures

- This is concerned with the division of responsibilities.
  - Between system components (apps, servers, processes)
  - Placement on computers in the network
- Implications for performance, reliability, and security.
- Types
  - Client-server model
  - Services provided by multiple servers
  - Proxy servers and caches
  - Peer processes
  - Mobile code / agents / spontaneous networking
  - Networked computers / thin clients
Client/Server Model, Multiple Servers

- We’re all familiar with this one. The web is the most widespread with browsers (clients) and web servers (servers).
- The model defines the interaction relationship.
  - Service: A task a machine can perform
  - Server: A machine that performs that task when requested
  - Client: A machine that requests the service

- The model allows chaining and hierarchy
  - Servers may be clients of other servers.
    - Example: WWW server using files provided by a file server.

- Service types
  - Directory service, print service, file service, …
Client/Server Model, Multiple Servers

- Services may be implemented by distributed processes.
  - May require distributed resources (such as the WWW)
  - May choose to partition and distribute for reliability
- Replication can be used to:
  - Increase performance
  - Increase availability
  - Improve fault tolerance
Clients Invoke Individual Servers

Diagram showing the interaction between clients and servers.
A Service Provided by Multiple Servers

- Example: load balancing very heavily used web servers by delegating clients to different servers based on individual server load or client proximity.
More on Client/Server Model

- **Clients**
  - Generally block until server responds or a timeout occurs.
  - Typically invoked by end users when they require service.
  - Interacts with users through a user interface.
  - Interacts with client middleware through middleware API to abstract above underlying network connectivity to server.

- **Server**
  - Implements services.
  - Usually waits for incoming requests.
  - Usually a program with special privileges.
  - Invoked by server middleware.
  - Provides error recovery and failure handling services.