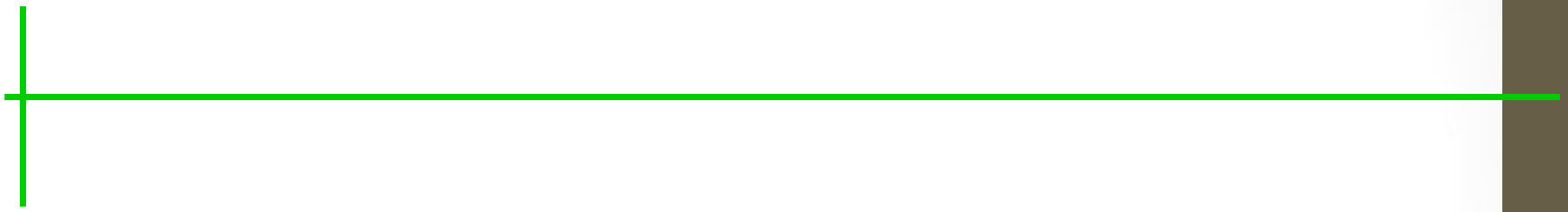
Lecture 1
Introduction to Distributed Systems



Presentation Outline

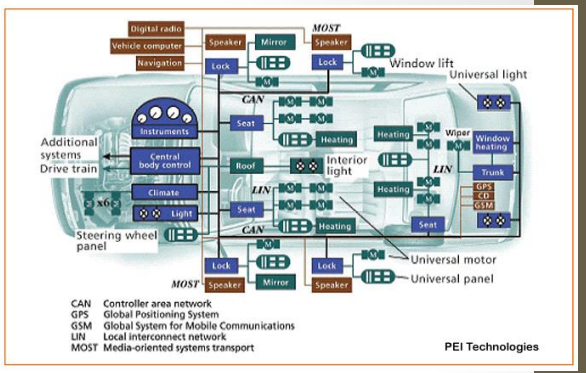
- Introduction
- Defining Distributed Systems
- Characteristics of Distributed Systems
- Example Distributed Systems
- Challenges of Distributed Systems
- Summary

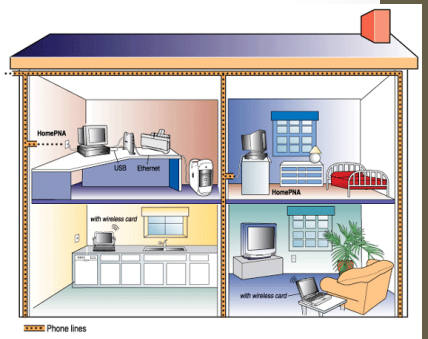
Aims of this module

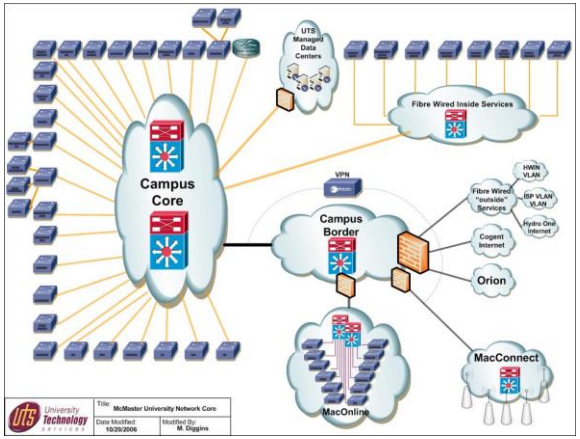
- Introduce the features of Distributed Systems that impact system designers and implementers
- Introduce the main concepts and techniques that have been developed to help in the tasks of designing and implementing Distributed Systems

Introduction

- Networks of computers are everywhere!
  - Mobile phone networks
  - Corporate networks
  - Factory networks
  - Campus networks
  - Home networks
  - In-car networks
  - On board networks in planes and trains







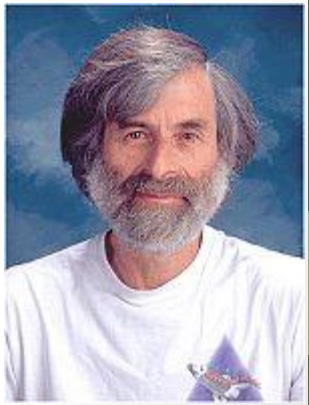
Defining Distributed Systems

- *"A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing."* [Coulouris]

- *"A distributed system is a collection of independent computers that appear to the users of the system as a single computer."* [Tanenbaum]

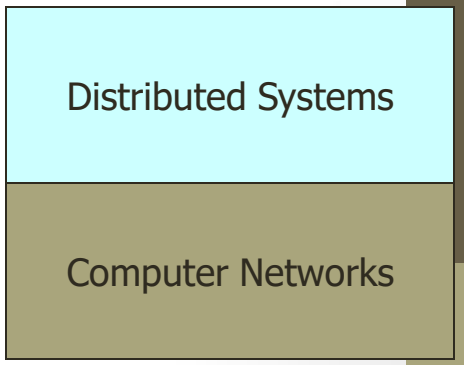
Leslie Lamport's Definition

- *"A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed."*
  - Leslie Lamport – a famous researcher on timing, message ordering, and clock synchronization in distributed systems.



Networks vs. Distributed Systems

- Networks: A media for interconnecting local and wide area computers and exchange messages based on protocols. Network entities are visible and they are explicitly addressed (IP address).
- Distributed System: existence of multiple autonomous computers is transparent
- However,
  - many problems (e.g., openness, reliability) in common, but at different levels.
    - Networks focuses on packets, routing, etc., whereas distributed systems focus on applications.
    - Every distributed system relies on services provided by a computer network.



Reasons for having Distributed Systems

- Functional Separation:
  - Existence of computers with different capabilities and purposes:
    - Clients and Servers
    - Data collection and data processing
- Inherent distribution:
  - Information:
    - Different information is created and maintained by different people (e.g., Web pages)
  - People
    - Computer supported collaborative work (virtual teams, engineering, virtual surgery)
  - Retail store and inventory systems for supermarket chains)

Reasons for having Distributed Systems

- Power imbalance and load variation:
  - Distribute computational load among different computers.
- Reliability:
  - Long term preservation and data backup (replication) at different locations.
- Economies:
  - Sharing resources to reduce costs and maximize utilization (e.g. network printer)
  - Building a supercomputer out of a network of computers.

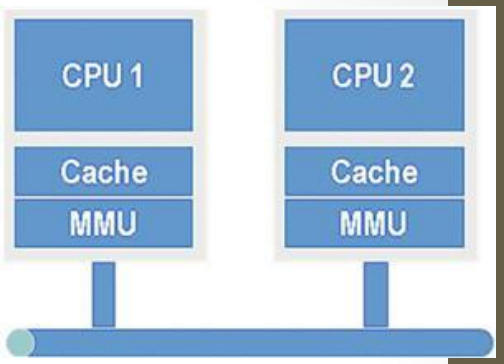
Characteristics of Distributed Systems

- Concurrency
  - Carry out tasks independently and parallely
  - Tasks coordinate their actions by exchanging messages
- Communication via message passing
  - No shared memory
- Resource sharing
  - Printer, database, other services
- No global state
  - No single process can have knowledge of the current global state of the system

Characteristics of Distributed Systems

- Heterogeneity – Different devices operating together
- Independent and distributed failures
- No global clock
  - Only limited precision for processes to synchronize their clocks

Differentiation with parallel systems



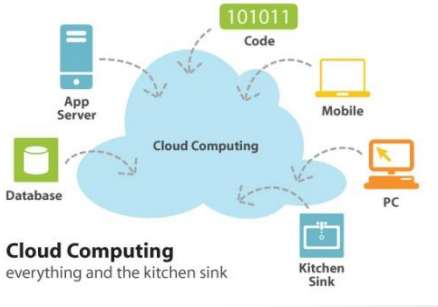
- Multiprocessor/Multicore systems
  - Shared memory
  - Bus-based interconnection network
  - E.g. SMPs (symmetric multiprocessors) with two or more CPUs, GPUs
- Multicomputer systems / Clusters
  - No shared memory
  - Homogeneous in hard- and software
    - Massively Parallel Processors (MPP)
      - Tightly coupled high-speed network
    - PC/Workstation clusters
      - High-speed networks/switches-based connection.

Differentiation with parallel systems is blurring

- Extensibility of clusters leads to heterogeneity
  - Adding additional nodes as requirements grow
- Leading to the rapid convergence of various concepts of parallel and distributed systems

Examples of Distributed Systems

- They (DS) are based on familiar and widely used computer networks:
  - Internet
  - Intranets, and
  - Wireless networks
- Example DS:
  - Web (and many of its applications like Facebook)
  - Data Centers and Clouds
  - Mobile applicaations
  - Wide area storage systems
  - Banking Systems



Challenges with Distributed Systems

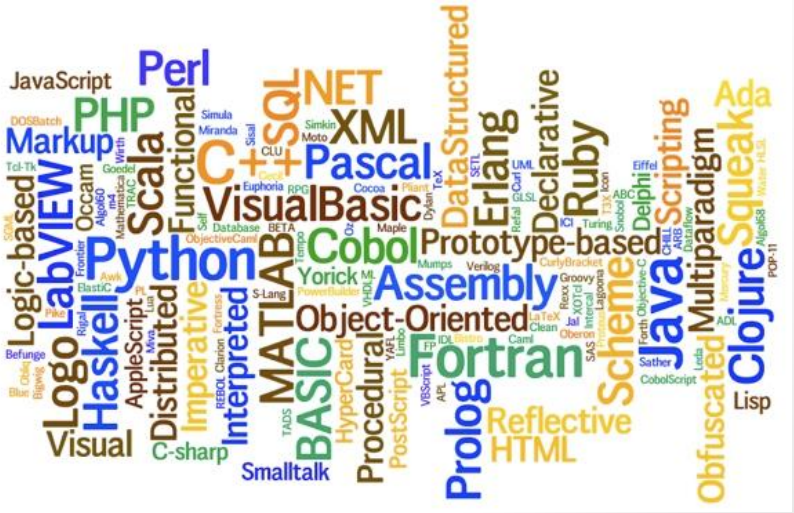
- Heterogeneity
  - Heterogeneous components must be able to interoperate
- Distribution transparency
  - Distribution should be hidden from the user as much as possible
- Fault tolerance
  - Failure of a component (partial failure) should not result in failure of the whole system
- Scalability
  - System should work efficiently with an increasing number of users
  - System performance should increase with inclusion of additional resources

Challenges with Distributed Systems

- Concurrency
  - Shared access to resources must be possible
- Openness
  - Interfaces should be publicly available to ease inclusion of new components
- Security
  - The system should only be used in the way intended

Heterogeneity

- Heterogeneous components must be able to interoperate across different:
  - Operating systems
  - Hardware architectures
  - Communication architectures
  - Programming languages
  - Software interfaces
  - Security measures
  - Information representation



Distribution Transparency

- To hide from the user and the application programmer the separation/distribution of components, so that the system is perceived as a whole rather than a collection of independent components.

- ISO Reference Model for Open Distributed Processing (ODP) identifies the following forms of transparencies:

- **Access transparency**
  - Access to local or remote resources is identical
  - E.g. Network File System / **Dropbox**
- **Location transparency**
  - Access without knowledge of location
  - E.g. separation of domain name from machine address.

.

Distribution Transparency II

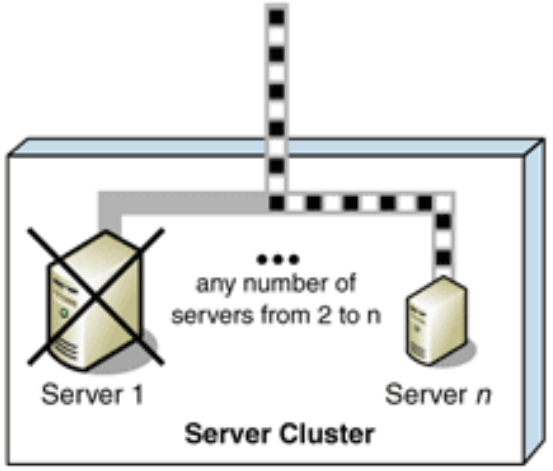
- Failure transparency
  - Tasks can be completed despite failures
  - E.g. message retransmission, failure of a Web server node should not bring down the website
- Replication transparency
  - Access to replicated resources as if there was just one. And provide enhanced reliability and performance without knowledge of the replicas by users or application programmers.
- Migration (mobility/relocation) transparency
  - Allow the movement of resources and clients within a system without affecting the operation of users or applications.
  - E.g. switching from one name server to another at runtime; migration of an agent/process from one node to another.

Distribution Transparency III

- **Concurrency transparency**
  - A process should not notice that there are other sharing the same resources
- **Performance transparency:**
  - Allows the system to be reconfigured to improve performance as loads vary
  - E.g., dynamic addition/deletion of components, switching from linear structures to hierarchical structures when the number of users increases
- **Scaling transparency:**
  - Allows the system and applications to expand in scale without changes in the system structure or the application algorithms.
- **Application level transparencies:**
  - Persistence transparency
    - Masks the deactivation and reactivation of an object
  - Transaction transparency
    - Hides the coordination required to satisfy the transactional properties of operations

Fault Tolerance

- Failure: an offered service no longer complies with its specification
- Fault: cause of a failure (e.g. crash of a component)
- Fault tolerance: no failure despite faults



Fault Tolerance Mechanisms

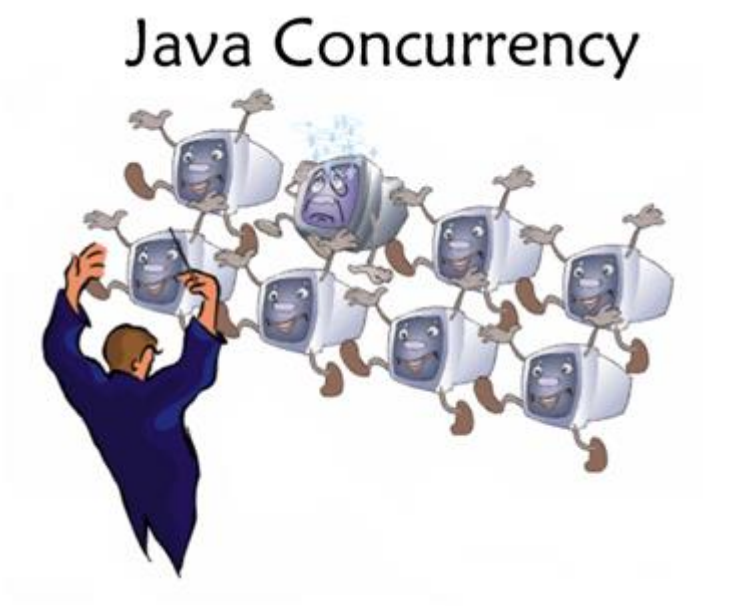
- Fault detection
  - Checksums, heartbeat, …
- Fault masking
  - Retransmission of corrupted messages, redundancy, …
- Fault toleration
  - Exception handling, timeouts,…
- Fault recovery
  - Rollback mechanisms,…

Scalability

- System should work efficiently at many different scales, ranging from a small Intranet to the Internet
- Remains effective when there is a significant increase in the number of resources and the number of users
- Challenges of designing scalable distributed systems:
  - Cost of physical resources
  - Performance Loss
  - Preventing software resources running out:
    - Numbers used to represent Internet addresses (32 bit->64bit)
    - Y2K-like problems
  - Avoiding performance bottlenecks:
    - Use of decentralized algorithms (centralized DNS to decentralized)

Concurrency

- Provide and manage concurrent access to shared resources:
  - Fair scheduling
  - Preserve dependencies (e.g. distributed transactions)
  - Avoid deadlocks
  - Preserve integrity of the system



Openness and Interoperability

- Open system:
  "... a system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software to be ported across a wide range of systems with minimal changes, to interoperate with other applications on local and remote systems, and to interact with users in a style which facilitates user portability" (Guide to the POSIX Open Systems Environment, IEEE POSIX 1003.0)

Openness and Interoperability

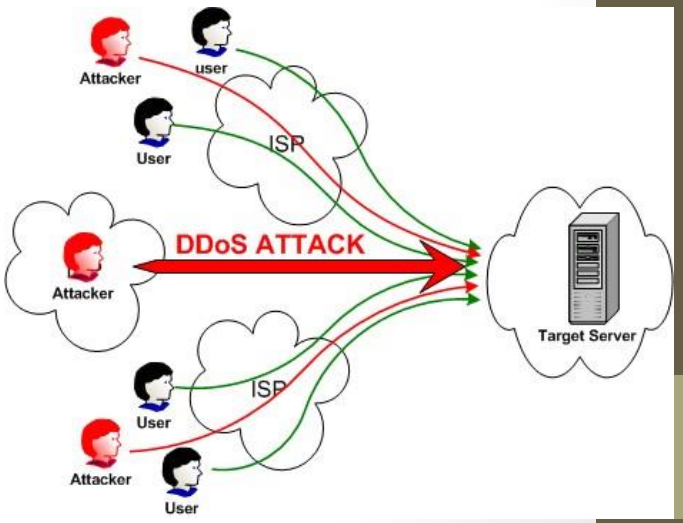
- Open message formats: e.g. XML
- Open communication protocols: e.g. HTTP, HTTPS
- Open spec/standard developers - communities:
  - ANSI, IETF, W3C, ISO, IEEE, OMG, Trade associations,...

Security I

- Resources are accessible to authorized users and used in the way they are intended
- Confidentiality
  - Protection against disclosure to unauthorized individual information
  - E.g. ACLs (access control lists) to provide authorized access to information
- Integrity
  - Protection against alteration or corruption
  - E.g. changing the account number or amount value in a money order

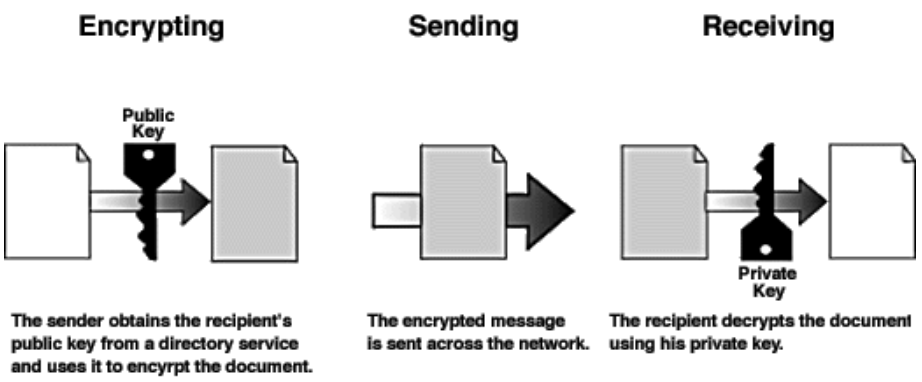
Security II

- Availability
  - Protection against interference targeting access to the resources.
  - E.g. denial of service (DoS, DDoS) attacks
- Non-repudiation
  - Proof of sending / receiving an information
  - E.g. digital signature



Security Mechanisms

- Encryption
  - E.g. Blowfish, RSA
- Authentication
  - E.g. password, public key authentication
- Authorization
  - E.g. access control lists



Business Example and Challenges

- Web/Mobile app to search and purchase online courses
  - Customers can connect their computer to your server (locally hosted or cloud hosted):
    - Browse your courses
    - Purchas courses
    - …

This example has been adapted from Torbin Weis, Berlin University of Technology

Business Example – Challenges I

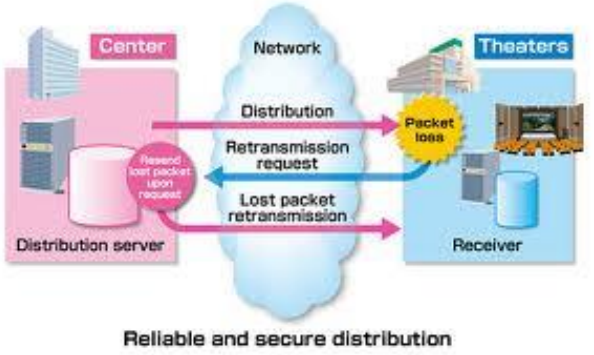
- What if
  - Your customers use different devices? (Dell laptop, Android device …)
  - Your customers use different OSs? (Android, IoS, Ubuntu…)
  - … … a different way of representing data? (Text, Binary,…)
  - **Heterogeneity**
- Or
  - You want to move your business and computers to China (because of the lower costs)?
  - Your client moves to a different country(more likely)?
  - **Distribution transparency**

Business Example – Challenges II

- What if
  - Two customers want to order the same item at the same time?
  - **Concurrency**
- Or
  - The database with your inventory information crashes?
  - Your customer's computer crashes in the middle of an order?
  - **Fault tolerance**

Business Example – Challenges III

- What if
  - Someone tries to break into your system to alter data?
  - … sniffs for information?
  - … someone says they have enrolled to the course but they haven't?
  - **Security**
- Or
  - You are so successful that millions of people are using your app at the same time.
  - **Scalability**



Business Example – Challenges IV

- When building the system…
  - Do you want to write the whole software on your own (network, database,…)?
  - What about updates, new technologies?
  - Adding a web client later on?
  - Will your system need to communicate with existing systems (e.g. payment gateways, SMS servers)
  - **Reuse** and **Openness** (Standards)

Impact of Distributed Systems

- New business models (e.g. Uber, Airbnb)
- Global financial markets
- Global labor markets
- E-government (decentralized administration)
- Ecommerce
- Driving force behind globalization
- Social/Cultural impact
- Media getting decentralized

Summary

- Distributed Systems are everywhere
- The Internet enables users throughout the world to access its services wherever they are located
- Resource sharing is the main motivating factor for constructing distributed systems
- Construction of DS produces many challenges:
  - Heterogeneity, Openness, Security, Scalability, Failure handling, Concurrency, and Transparency
- Distributed systems enable globalization:
  - Community (Virtual teams, organizations, social networks)
  - Science (e-Science)
  - Business (e-Bussiness)