Introduction

Distributed Systems

Instructor: Michael Mutisya

mutisyamike07@gmail.com

0720752707

Introduction

- Invention of high-speed computer networks
  - Local-area networks (LANs)
    - Small amount of information, a few microseconds
    - Large amount of information, at rate of 100 million to 10 billion bits/sec
  - Wide-area networks (WANs)
    - 64 Kbps to gigabits per second
- Consequently, it is *easy* to connect computer systems via high speed networks

DS: Definition (Tanenbaum and van Steen)

- A distributed system is a piece of software that ensures that
  - *A collection of independent computers appears to its users as a single coherent system.*

Distributed System:

- **Two aspects:**

  (1) independent computers and

  (2) single system ⇒ middleware

  - **Middleware** is software that lies between an operating system and the applications running on it. Essentially functioning as hidden translation layer, **middleware** enables communication and data management for distributed applications

DS Definition: Lamport

- "You know you have one when the crash of a computer you've never heard of stops you from getting any work done."
  - *A distributed system is one in which I cannot get something done because a machine I've never heard of is down.*

Application Domains

- Finance and commerce – ebay, amazon, paypal
- Information Society –www, search engines, digital libraries
- Creative industries and entertainment – online games(MMOG)

Characteristics of a DS

- Multiple computers
  - Concurrent execution
  - Independent operation and failures
- Communications
  - Ability to communicate
  - No tight synchronization
- Relatively easy to expand or scale
- Transparency

DS: Motivation

- Economics
  - Share resources
  - Relatively easy to expand or scale
  - Speed – A distributed system may have more total computing power than a mainframe.
  - Cost
- Location Independence
- People and information are distributed
- Availability and Reliability
  - If a machine crashes, the system as a whole can survive.

Distributed System Design Goals

1. Making resources available

- The main goal of a distributed system is to make it easy for users to access remote resources and to share them with others in a controlled way.
- It is cheaper to le a printer be shared by several users than buying and maintaining printers for each user.
- Collaborating and exchanging information can be made easier by connecting users and resource.

Distributed System Design Goals cont'd

2. Distribution Transparency:

- It is important for a distributed system to hide the location of its process and resource. A distributed system that can portray itself as a single system is said to be transparent.
- The various transparencies need to be considered are access, location, migration, relocation, replication, concurrency, failure and persistence.
- Aiming for distributed transparency should be considered along with performance issues.

Distributed System Design Goals cont'd

3. Openness:

- Openness is an important goal of distributed system in which it offers services according to standard rules that describe the syntax and semantics of those services.
- Open distributed system must be flexible making it easy to configure and add new components without affecting existing components.
- An open distributed system must also be extensible.

Distributed System Design Goals cont'd

4. Scalable:

- Scalability is one of the most important goals which are measured along three different dimensions.
- First, a system can be scalable with respect to its size which can add more user and resources to a system.
- Second, users and resources can be geographically apart.
- Third, it is possible to manage even if many administrative organizations are spanned.

Distribution Transparency

- Transparency
  - This is hiding the fact that its processes and resources are physically distributed across multiple computers
  - A DS that is able to present itself to users as if it were only on a single computer system is said to be transparent
  - Note
    - Distribution transparency is a nice a goal, but achieving it is a different story.

Distribution Transparency

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Degree of Transparency

- Aiming at full distribution transparency may be too much:
  - Users may be located in different continents
  - Completely hiding failures of networks and nodes is (theoretically and practically) impossible
    - You cannot distinguish a slow computer from a failing one
    - You can never be sure that a server actually performed an operation before a crash
- Full transparency will cost performance, exposing distribution of the system
  - Keeping Web caches exactly up-to-date with the master
  - Immediately flushing write operations to disk for fault tolerance

Openness of Distributed Systems

- Goal: Open distributed system -- able to interact with services from other open systems, irrespective of the underlying environment:
- Standard rules (protocols/interfaces) to describe services/components
- Interfaced definitions should be Complete & Vendor neutral
- These help making system / services interoperable & portable
  - Flexibility – ability to integrate multiple components

Achieving openness

- Achieving openness: At least make the distributed system independent from the underlying environment:
  - Hardware
  - Platforms
  - Languages

Scale in Distributed Systems

- Observation: Many developers of modern distributed system easily use the adjective "scalable" without making clear why their system actually scales.
- Three metrics of a scalable system:
  - Number of users and/or processes (size scalability)
  - Maximum distance between nodes (geographical scalability)
  - Number of administrative domains (administrative scalability)- used IDs, user groups, ACLs, resource managers

Scalability Limitations

- Centralized Services: Single server for all users. Often necessary.
- Centralized Data: Single online telephone book.
- Centralized Algorithms: Doing routing based on complete information.

Scalability Issues

- Decentralized algorithms:
  - No machine has complete information about the system state.
  - Machines make decisions based only on local information.
  - Failure of one machine does not ruin the algorithm

Scalability Issues cont'd

- **Other issues with Geographical scalability**
  - **Problems due to Synchronous communication-** sender will wait until the acknowledgment is received from the receiver and receiver waits until the message arrives.
  - **Unreliable WANs.**
- **Scaling the system across multiple independent administrative domains.**
  - **Conflicting policies w.r.t to resource usage (payment), management and security.**

Scaling Techniques

1. Partition data and computations across multiple machines
   - Move computations to clients (Java applets)
   - Decentralized naming System (DNS)
   - Decentralized information systems (WWW)
2. Make copies of data available at different machines (Servers)
   - Replicated file servers (for fault tolerance)
   - Replicated databases
   - Mirrored web sites

DNS Structure

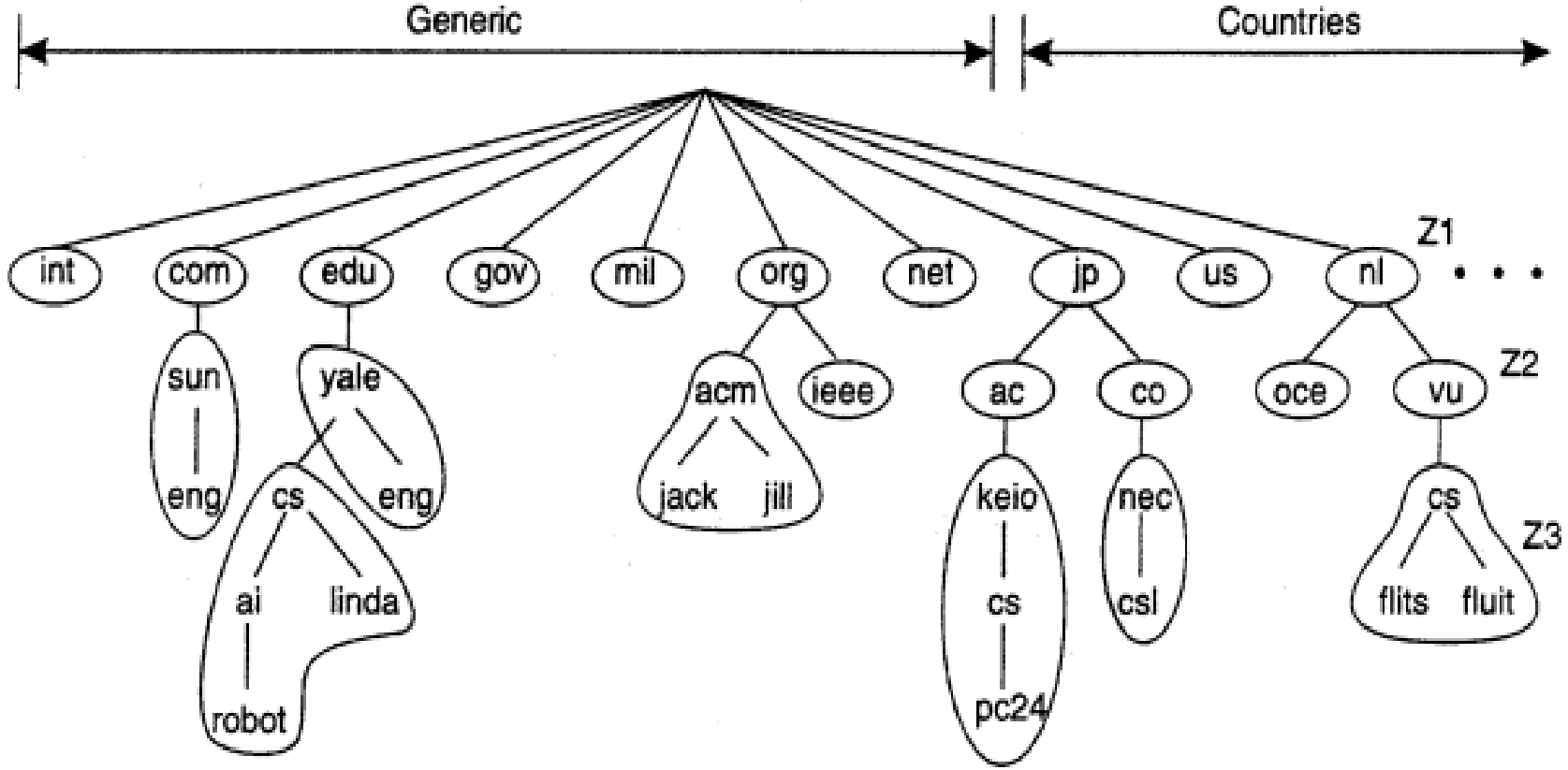

Figure 1-5. An example of dividing the DNS name space into zones.

Scaling Techniques cont'd

3. Allow client processes to access local copies

- Attempts to reduce the network traffic of the previous model by caching the data obtained from the server node
- **Web caches (browser/Web proxy)**
- **File caching (at server and client)**

4. Hiding communication latencies

- Important to achieving geographical scalability- try to avoid waiting for responses to remote service requests
- **Reduce overall communication for interactive applications.**

Developing Distributed Systems: Pitfalls

- **Observation: Many distributed systems are needlessly complex caused by mistakes that required patching later on.**
- **Many possible false assumptions:**
  - **The network is reliable**
  - **The network is secure**
  - **The network is homogeneous-** where all the nodes have the same function in the **network**
  - **The topology does not change**

Developing Distributed Systems: Pitfalls cont'd

- Bandwidth is infinite
- There is one administrator

- A common misconception among people when discussing distributed systems is that it is just another name for a network of computers. However, this overlooks an important distinction.
- A distributed system is built on top of a network and tries to hide the existence of multiple autonomous computers. It appears as a single entity providing the user with whatever services are required.

Distributed System Challenges

1. Heterogeneity

- Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another

- **networks;**
- **computer hardware;**
- **operating systems;**
- **programming languages;**
- **implementations by different developers.**

Distributed System Challenges cont'd

2. Openness

- The openness of distributed systems is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs.
- If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future. Example: Twitter has API that allows developers to develop theirs own software interactively.

Distributed System Challenges cont'd

3. Scalability

Distributed System Challenges

4. Security
   - Confidentiality - protection against disclosure to unauthorized individuals
   - Integrity - protection against alteration or corruption
   - Availability - protection against interference with the means to access the resources
   - Firewalls, encryption techniques
- Challenges - denial of service attacks, security of mobile code (exes as attachments)

Distributed System Challenges

5. Failure Handling
   - Detecting failures e.g. checksums for corrupt data
   - Masking failures e.g. retransmissions
   - Recovery from failures
   - Redundancy e.g. use of RAID
- Concurrency
- Transparency
- Quality of Service(QoS)