Introduction to Distributed Systems

February 14, 2016

Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

Challenges

Further Reading

Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

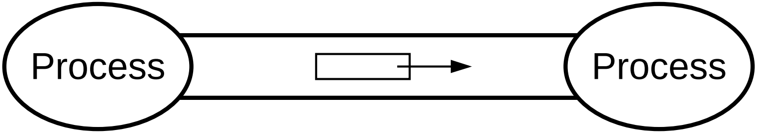
Challenges

Further Reading

Distributed System

Definition A distributed system consists of a **collection of** distinct **processes** which are spatially separated and **which communicate with one another by exchanging messages**.

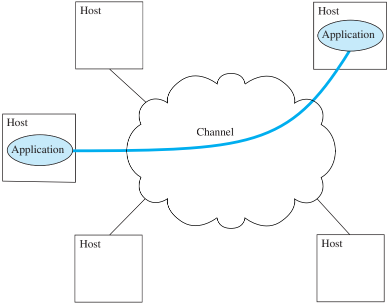
- "A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process." (L. Lamport, "Time, Clocks and the Order of Events in a Distributed System", CACM)



Message based communication

Message a sequence of bits

- Whose format and meaning are specified by a *communication a protocol*
- That is transported from its source to its destination by a **communications network**



Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

Challenges

Further Reading

Examples?

Other Distributed Systems/Applications.

- Web and Internet
- Google search service
- *Email service*
- *Peer-to-peer* applications, such as Bittorrent
- FEUP's file system
- Telecommunication networks
- ATM networks (SIBS)
- Home automation
- Factory automation
- Fly-by-wire, drive-by-wire.

Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

Challenges

Further Reading

Potencial Advantages

- Sharing of resources
- Access to remote resources
- Performance
  - Can use multiple computers to solve a problem
- Scalability:
  - Load (no. of users/request rate)
  - Geographical;
  - Administrative
- Fault tolerance
  - Reliability
  - Availability

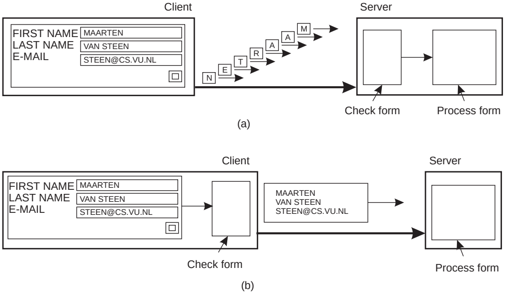
Scalability: Challenges

- Centralization
    - processing;
    - data;
    - algorithms.
- Synchronous/blocking comunication
- Security and (lack of) trust

Scalability: Some Techniques (1/2)

Distribuition

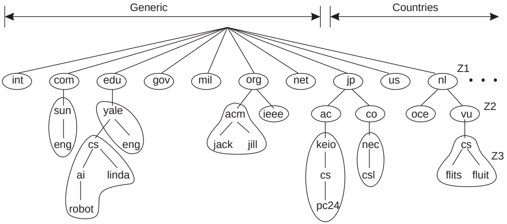
processing:



(a)

(b)

data (partitioning):



Scalability: Some Techniques (2/2)

- Distributed (decentralized) algorithms:
    - System global state is unknown (relativity)
        - Can use only information locally available
    - Correctness must be ensured even in the presence of faults
    - No single physical clock
- Asynchronous/non-blocking communication
- Replication and *caches*:
    - + reduces communication latency;
    - + allow distributed processing;
    - - raises consistency problems

Fault Tolerance: Challenges and Techniques

Challenges

- Distributed systems may exhibit partial-failure modes
- Fault detection is not always possible
  - How do you distinguish a slow computer from a crashed computer?
  - How do you know if a server has executed an action before crashing?
- Distributed consensus in the presence of faults is not always possible (in theory, and ... in practice)
  - Actually, this is a corollary of the previous on ailure detection

Tecnhiques

- Transactions
- Replication

Summary

- What is a distributed system?
- Examples
- Why distribution?
- Design Goals
- Challenges
- Further Reading

Design Goals

Distribution transparency

- “Hide” from users (and sometimes from programmers) the distributed nature of the application

Openness understood as the possibility

1. of implementing the application/system in different ways
   - this is a common goal of SW engineering
2. of expanding the application/system functionality
   - this is particularly relevant for service oriented architectures

Distribution Transparency Facets

location access without knowledge of the resource location;

concurrency access as if the resources were not shared;

fault access to resources even in the presence of faults;

replication access as if there was a single instance of the resource

- There are at least two levels of transparency:
  - with respect to users;
  - with respect to developers/programmers.

Degree of Distribution Transparency

- *Too much of a good thing can be bad*
- Sometimes you **do not want** to hide the location of a resource:
  - For example, consider a distributed printing application
- Sometimes you **cannot** hide faults in the network or in computers (both in theory and in practice):
  - It is not always possible to distinguish a slow computer from a crashed computer
  - It is not always possible to learn if an operation was executed before the crash of a server
    - Even after recovery

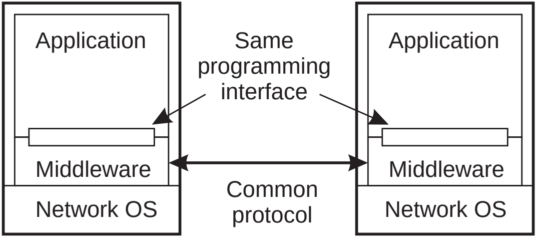
Openess and Middleware

Interfaces must be public and well defined

- Ensures that the code is portable

Protocols must be standard

- Ensures interoperability



Middleware

- Allows an application to be independent of:
  - the platform (HW + OS)
  - the programming language (not always)

Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

Challenges

Further Reading

Challenges

- Partial failures
  - Some components may fail, while others continue to operate correctly
- IPC latency
  - IPC across the network has a larger and unpredictable latency, which usually cannot be bounded
- No global time
- No memory access
  - Pointers are meaningful only in the context of the respective address space
- Heterogeneity
  - Has several facets
- Lack of security and trust

Distributed Computing Fallacies (Sun People)

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. The topology does not change.
6. There is one administrator.
7. Transport cost (in $) is zero.
8. The network is homogeneous.

The word **network** should not be interpreted in its narrower meaning.

Summary

What is a distributed system?

Examples

Why distribution?

Design Goals

Challenges

Further Reading

Further Reading

- *Distributed Systems, 2nd Ed.*, Chapter 1
- Michael Schroeder (et. al.) State-of-the-Art Distributed System: Computing with BOB
  - Nice "vision" from leading distributed system's researchers of DEC's SRC around 1990
  - Read only Sections 1 and 2
- Jim Waldo, et. al, A Note on Distributed Computing
  - Somewhat language-oriented, by people who designed Java RMI
- Arnon Rotem-Gal-Oz, Fallacies of Distributed Computing Explained
  - Too much hype, almost ... "evangelist" style
- Stephen Asbury, The Eight Fallacies of Distributed Computing
  - An entertaining talk.