Contents

Contributors:

- Daniel Fitz (Sanchez)

0.1 Assessment

- Final Exam (50%)
  - All goals tested except for developing applications
  - **Closed book exam**
- Two assignments
  - Individual programming assignment (25%)

    Implementation of a context-aware distributed application using RMI and “publish/subscribe” (event based architecture)
  - Group research/written assignment (25%)

    Group assignment on functionality and design issues of various distributed systems (e.g. grid computing, cloud computing, pervasive computing)

Chapter 1

Lecture Notes

1.1 Introduction to Distributed Systems

1.1.1 Definitions of Distributed Systems

> A collection of independent computers that appear to its users as a single coherent system (Andrew Tannenbaum)

> A system where I can't get my work done because a computer has failed that I've never even heard of (Leslie Lamport)

A distributed system is a collection of independent computers that are used jointly to perform a single task or to provide a single service.

Note 1: Characteristics

- Multiple computers
  CPU, memory, storage, I/O
- Interconnections
  variety of interconnection architectures
- Resources
  remote access to resources
  resource can be shared

1.1.2 Goals of Distributed Systems

- Transparency (hiding distribution)
  System presents itself as a single computer system
- Openness
  Interoperability, portability, heterogeneity
- Scalability
  Ability to grow

Transparency

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 is replicated
Concurrency: Hide that a resource may be shared by several competitive users
Failure: Hide the failure and recovery of a resource

Openness

- Interoperability
- Portability
- Heterogeneity
- Standard interfaces
- Interface Definition Language (IDL)

Scalability

' Three axis of scalability:
- Administratively
- Geographically
- Size (users, resources)

Algorithms vs Scalability Decentralized algorithms should be used:
- 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
- There is no implicit assumption that a global clock exists

Scaling Techniques
- Hiding communication latencies
  Asynchronous communication
  Client-side processing
- Distribution
  Split and spread functionality across the system
  Decentralize algorithms
- Replication (including caching)

If asynchronous communication cannot be used - communication should be reduced

1.1.3 Types of Distributed Systems

Distributed Computing Systems

- Cluster Computing Systems

  Just a bunch of computers all connected over a shared network
- Grid Computing Systems

  Layered System: Applications → Collective Layer → (Connectivity layer / Resource layer) → Fabric layer
- Cloud Computing

  Paradigm for enabling **network access** to a scalable and elastic pool of **shareable physical or virtual resources** with on-demand self-service provisioning and administration

Distributed Information Systems

- Transaction processing systems

  There are many information systems in which many distributed operations on (possibly distributed) data have to have the following behavior (either all of the operations are executed, or none of them is executed):

  BEGIN_TRANSACTION: Mark the start of a transaction

  END_TRANSACTION: Terminate the transaction and try to commit

  ABORT_TRANSACTION: Kill the transaction and restore the old values

  READ: Read data from a file, a table, or otherwise

  WRITE: Write data to a file, a table, or otherwise

Note 2: Distributed Transactions - Model

A transaction is a collection of operations that satisfies the following ACID properties:

Atomicity: All operations either succeed, or all of them fail. When the transaction fails, the state of the object will remain unaffected by the transaction.

Consistency: A transaction establishes a valid state transaction. This does not exclude the possibility of invalid, intermediate states during the transaction's execution.

Isolation (Serialisability): Concurrent transaction do not interfere with each other. It appears to each transaction *T* that other transactions occur either *before T*, or *after T*, but never both.

Durability: After the execution of a transaction, its effects are made permanent: changes to the state survive failures.

- Enterprise application integration

  Middleware as a communication facilitator in enterprise application integration

  Multiple applications communicate to the middleware which then talks to all the server-side applications

Distributed Pervasive Systems

Pervasive systems:

- Embedded devices
- Mobile devices
- Heterogeneous networks
- (Autonomic) Adaptation to context changes

  Adaptation to changes in the infrastructure

  Adaptation to user tasks/needs

Requirements for pervasive systems:

- Embrace contextual changes
- Encourage ad hoc composition
- Recognize sharing as the default

Home Systems (Smart Homes) Integration of entertainment and appliances into an "intelligent" adaptive system. May include health-monitoring and also provide support for independent living of the elderly.

Sensor Networks There is a variety of sensor networks, e.g.

- A small set of sensors supporting smart home

- A network of thousands of sensors providing climate monitoring

1.1.4 Architectures of Distributed Systems

Architecture styles

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures

System architectures

(how software components are distributed on machines)

- Centralized architectures (client-server: two-tiered, three-tiered, N-tiered)
  - The simplest organization is to have only two types of machines:
    - * A client machine containing only the programs implementing (part of) the user-interface level
    - * A server machine containing the rest

      the programs implementing the processing and data level
- Decentralized architectures (peer-to-peer)
  - Overlay network is constructed in a random way
  - Each node has a list of members but the list is created in unstructured (random) way
- Hybrid architectures (edge-server, collaborative DS)

  Clients participate in providing services: e.g. file sharing, when part of file is downloaded it's seeded to other clients

Note 3: Application Layering

- The user-interface level
- The processing level
- The data level