

Introduction to Distributed Systems (DS)

INF5040/9040 autumn 2009

lecturer: Frank Eliassen

Frank Eliassen, Ifi/UiO

1

Outline

- What is a distributed system?
- Challenges and benefits of distributed system
- Distribution transparencies
- Types of distributed systems
- Pitfalls when developing distributed systems

Frank Eliassen, Ifi/UiO

2

What is a distributed system?

Many definitions

- [Coulouris & Emmerich]
 - A distributed system consists of hardware and software components located in a network of computers that communicate and coordinate their actions only by passing messages
- [Tanenbaum & van Steen]
 - A distributed system is a collection of independent computers that appears to its users as a single coherent system.
- [Lamport]
 - A distributed system is a system that prevents you from doing any work when a computer you have never heard about, fails.
- The above definitions take different perspectives
 - Operational perspective
 - User perspective
 - DS characteristics perspective

Frank Eliassen, Ifi/UiO

3

A distributed system organized as middleware

- Layer of software offering a single-system view
- Offers portability and interoperability
- Simplifies development of distributed applications and services



Distributed applications and services

Platform Independent API

DISTRIBUTION MIDDLEWARE

Platform Dependent API

Local OS
1

Local OS
2

...

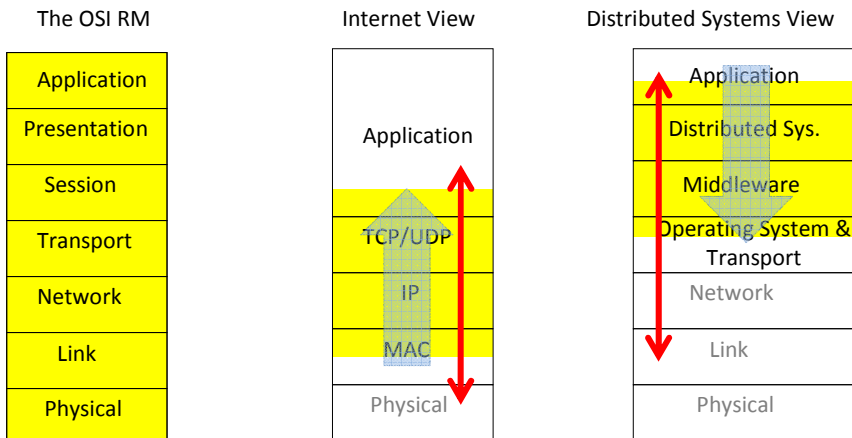
Local OS
n

- transaction oriented (ODTP XA)
- message oriented (IBM MQSeries)
- remote procedure call (X/Open DCE)
- object-based (CORBA, COM, Java)

Frank Eliassen, Ifi/UiO

4

Networks and distributed systems



Today: Cross-layering is applied in both worlds
Frank Eliassen, Ifi/UiO

5

Implications of distributed systems

- Independent failure of components
 - “partial failure” & incomplete information
- Unreliable communication
 - Loss of connection and messages. Message bit errors
- Unsecure communication
 - Possibility of unauthorised recording and modification of messages
- Expensive communication
 - Communication between computers usually has less bandwidth, longer latency, and costs more, than between independent processes on the same computer
- Concurrency
 - components execute in concurrent processes that read and update shared resources. Requires coordination
- No global clock
 - makes coordination difficult (ordering of events)

Frank Eliassen, Ifi/UiO

6

Goals of distributed systems

- resource sharing
 - the possibility of using available resources any where
- openness
 - an open distributed system can be extended and improved incrementally
 - requires publication of component interfaces and standards protocols and for accessing interfaces
- scalability
 - the ability to serve more users, provide acceptable response times with increased amount of data
- fault tolerance
 - maintain availability even when individual components fail
- allow heterogeneity
 - network and hardware, operating system, programming languages, implementations by different developers

Frank Eliassen, Ifi/UiO

7

Resource sharing

- The opportunity to use available hardware, software or data any where in the system
- Resource managers control access, offer a scheme for naming, and controls concurrency
- A resource manager is a software module that manages a resource of a particular type
- A resource sharing model describes how
 - resources are made available
 - resources can be used
 - service provider and user interact with each other

Frank Eliassen, Ifi/UiO

8

Models for resource sharing

- Client-server resource model
 - Server processes act as resource managers, and offer services (collection of procedures)
 - Client processes send requests to servers
 - HTTP defines a client-server resource model
- Object-based resource model
 - Any entity in a process is modeled as an object with a message based interface that provides access to its operations
 - Any shared resource is modeled as an object
 - Object based middlewares (CORBA, Java RMI) defines object-based resource models

Frank Eliassen, Ifi/UiO

9

Scalability

- A system is scalable if it remains effective when there is a significant increase in the amount of resources (data) and number of users
 - Internet: no of users and services has grown enormously
- Scalability denotes the ability of a system to handle an increasing future load
- Requirements of scalability often leads to a distributed system architecture (several computers)

Frank Eliassen, Ifi/UiO

10

Scalability problems (1)

- Often caused by centralized solutions

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Frank Eliassen, Ifi/UiO

11

Scalability problems (2)

- Characteristics of 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.
 - There is no implicit assumption that a global clock exists.

Frank Eliassen, Ifi/UiO

12

Scaling techniques (1)

➤ Distribution

- splitting a resource (such as data) into smaller parts, and spreading the parts across the system (cf DNS)

➤ Replication

- replicate resources (services, data) across the system
- increases availability, helps to balance load
- caching (special form of replication)

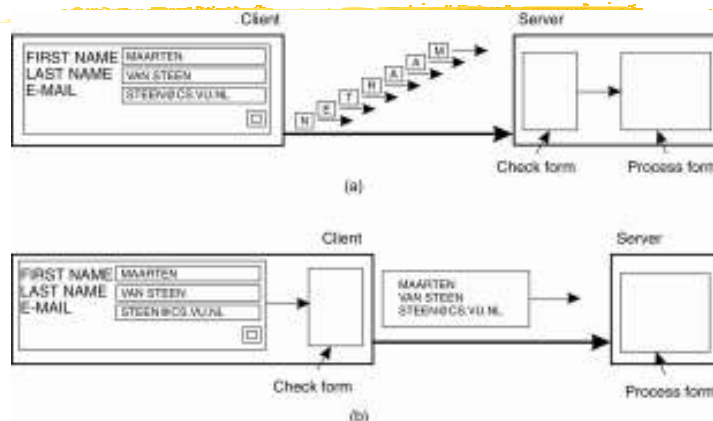
➤ Hiding communication latencies

- avoid waiting for responses to remote service requests (use asynchronous communication or design to reduce the amount of remote requests)

Frank Eliassen, Ifi/UiO

13

Scaling techniques (2)



- Reducing amount of remote requests: The difference between letting (a) a server or (b) a client check forms as they are being filled.

Frank Eliassen, Ifi/UiO

14

Failure handling

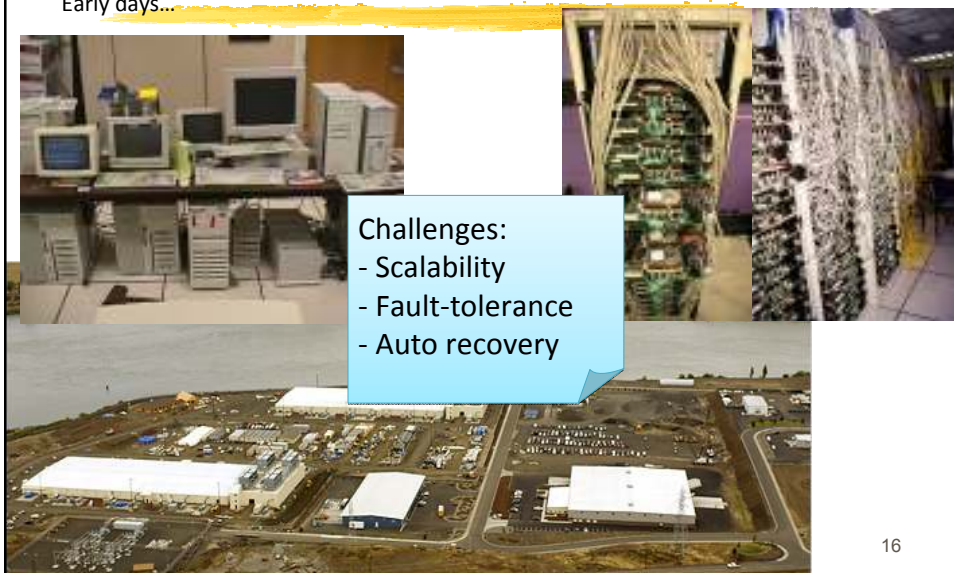
- Hardware, software and network fail!!
- DS must maintain availability even in cases where hardware/software/network have low reliability
- Failures in distributed systems are partial
 - makes error handling particularly difficult
- Many techniques for handling failures
 - Detecting failures (checksum a.o.)
 - Masking failures (retransmission in protocols)
 - Tolerating failures (as in web-browsers)
 - Recovery from failures
 - Redundancy (replicate servers in failure-independent ways)

Frank Eliassen, Ifi/UiO

15

Example: Google File-System

Early days...



16

Distribution transparency

- An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers
- A distributed system that is able to present itself to its users and applications as if it were only a single computer system is said to be **transparent**

Frank Eliassen, Ifi/UiO

17

Transparency in a distributed system

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

Different forms of transparency in a distributed system (ISO, 1995).

Trade-off between degree of transparency and performance of a system

Frank Eliassen, Ifi/UiO

18

Types of distributed system

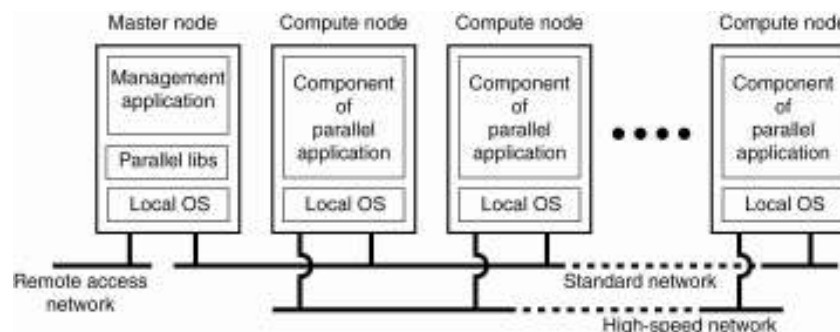
- Distributed Computing Systems
 - Used for high performance computing tasks
 - Cluster computing systems
 - Grid computing systems
- Distributed Information Systems
 - Systems mainly for management and integration of business functions
 - Transaction processing systems
 - Enterprise Application Integration
- Distributed Pervasive (or Ubiquitous) Systems
 - Mobile and embedded systems
 - Home systems
 - Sensor networks

Frank Eliassen, Ifi/UiO

19

Cluster Computing Systems

Collection of similar PCs, closely connected, all run same OS



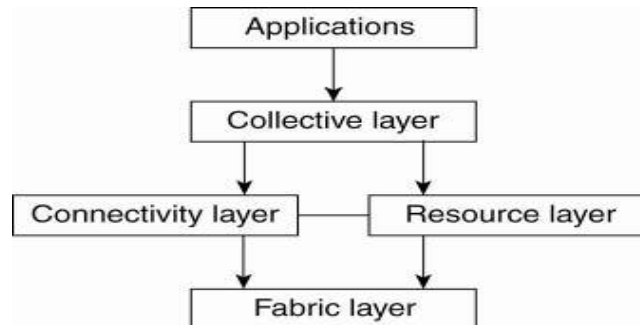
An example of a cluster computing system.

Frank Eliassen, Ifi/UiO

20

Grid Computing Systems

Federation of autonomous and heterogeneous computer systems (HW,OS,...), several adm domains



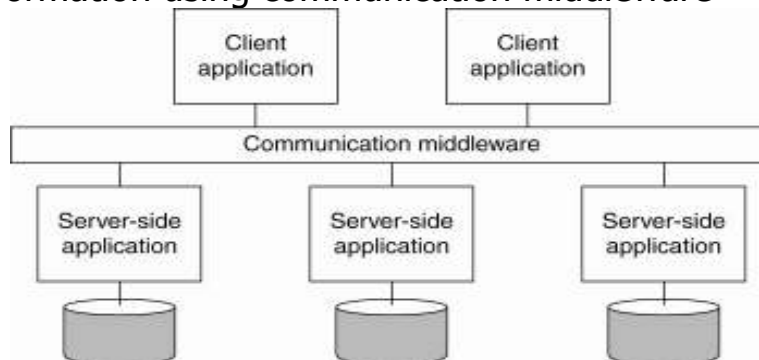
A layered architecture for grid computing systems.

Frank Eliassen, Ifi/UiO

21

Enterprise Application Integration

Allowing existing applications to directly exchange information using communication middleware



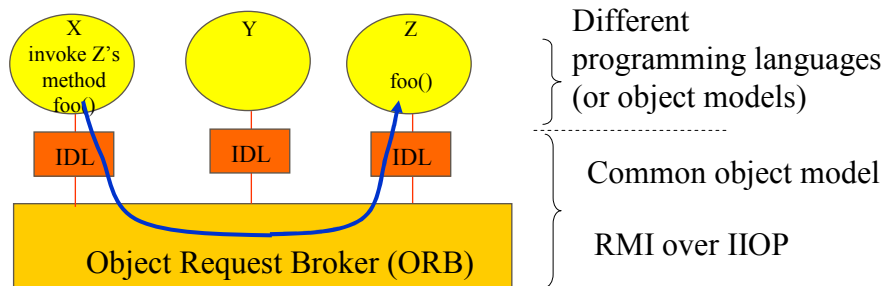
Middleware as a communication facilitator in enterprise application integration

Frank Eliassen, Ifi/UiO

22

Example communication middleware: CORBA

Clients may invoke methods of remote objects without worrying about:
object location, programming language,
operating system platform, communication
protocols or hardware.



Frank Eliassen, Ifi/UiO

23

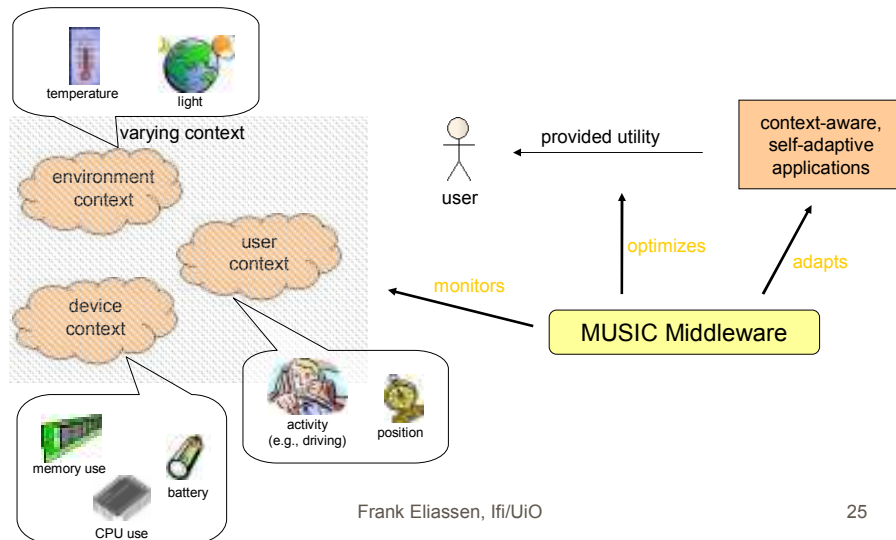
Distributed Pervasive Systems

- Devices in distributed pervasive systems discovers the environment and establishes themselves in this environment as best as possible.
- Requirements for pervasive applications
 - Embrace contextual changes.
 - Encourage ad hoc composition.
 - Recognize sharing as the default.

Frank Eliassen, Ifi/UiO

24

QUA/MUSIC context-aware adaptation middleware for distributed pervasive systems



25

Pitfalls when Developing Distributed Systems

- False assumptions made by first time developer:
 - The network is reliable.
 - The network is secure.
 - The network is homogeneous.
 - The topology does not change.
 - Latency is zero.
 - Bandwidth is infinite.
 - Transport cost is zero.
 - There is one administrator.

Frank Eliassen, Ifi/UiO

26

Summary

- Distributed systems:
 - components located in a network that communicates and coordinates their actions exclusively by sending messages.
- Consequences of distributed systems
 - Independent failure of components
 - Unsecure communication
 - No global clock
- Distribution transparency: providing a single computer system view
- Requirements like resource sharing, openness, scalability, fault tolerance and heterogeneity can be satisfied by distributed systems
- Many pitfalls when developing distributed systems