

Sorts of Distributed Systems

Distributed Systems

Sistemi Distribuiti

Andrea Omicini

`andrea.omicini@unibo.it`

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna a Cesena

Academic Year 2016/2017

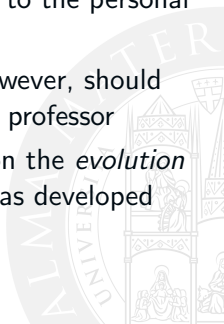


- 1 Distributed Computing Systems
- 2 Distributed Information Systems
- 3 Distributed Pervasive Systems



About these Slides

- the following slides borrow a great deal from the slides coming with the book adopted as the basic one for this course
[Tanenbaum and van Steen, 2007]
- material from those slides (including pictures) has been re-used in the following, and integrated with new material according to the personal view of the professor
- every problem or mistake contained in these slides, however, should be attributed to the sole responsibility of this course's professor
- the goal, here, is to be exposed to the classical view on the *evolution* of distributed systems that the scientific community has developed over the last decades



Evolution of Distributed Systems

Evolving technological and application scenarios...

- ... forced new requirements, structure, and goals for distributed systems
- resulting in the emergence of diverse classes of distributed systems over time



Sorts of Distributed Systems

Three main classes of distributed systems

- distributed computing systems
- distributed information systems
- distributed pervasive systems



Next in Line...

- 1 Distributed Computing Systems
- 2 Distributed Information Systems
- 3 Distributed Pervasive Systems



Distributed Computing Systems

The main characteristic

- using a *multiplicity of distributed computers* to perform *high-performance* tasks

Two classes

- **cluster** computing systems
- **grid** computing systems



Cluster Computing Systems I

The basic idea

- a collection of similar workstations / PCs
- running the same OS
- located in the same area
- interconnected through a high-speed LAN

Motivation

- the ever increasing price / performance ration of computers makes it cheaper to build a supercomputer by putting together many simple computers, rather than buying a high-performance one
- also, robustness is higher, maintenance and incremental addition of computing power is easier

Cluster Computing Systems II

Usage

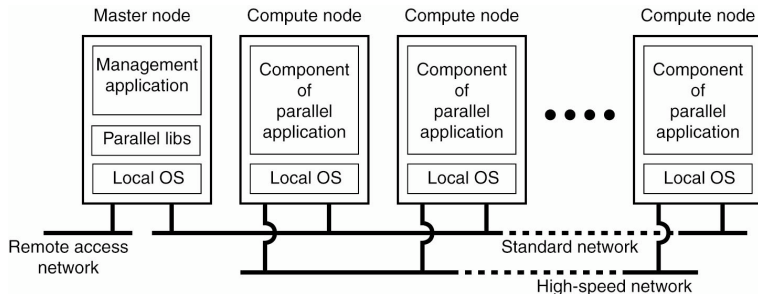
- parallel computing
- typically, a single computationally-intensive program is run in parallel on multiple machines



Cluster Computing Systems: Example

Beowulf clusters

- Linux-based
- each cluster is a collection of computing nodes controlled and accessed by a *single master node*



[Tanenbaum and van Steen, 2007]

Cluster vs. Grid Computing Systems

Homogeneity vs. heterogeneity

- homogeneity
 - computers in a cluster are typically similar
 - computers in a cluster have the same OS
 - computers in a cluster are connected to the same (local) network
- in essence, cluster computer systems are **homogeneous**
- grid computer systems instead are typically **heterogeneous**



Grid Computing Systems

The main idea

- resources from different organisations are brought together to promote *collaboration* between individuals, groups, or institutions, by passing organisation boundaries
- collaboration is built in the form of a *virtual organisation*
 - essentially, a new virtual organisational entity including people from existing organisations
 - accessing resources made available by participating organisations
 - including servers, databases, hard disks, . . .
- by their very nature, grid computer systems deal with different administrative domains

Architecture of a Grid Computing System I

A layered architecture for a grid computing system [Foster et al., 2001]

fabric layer — interface to local resources at a specific site

resource layer — management of single resources—e.g., access control

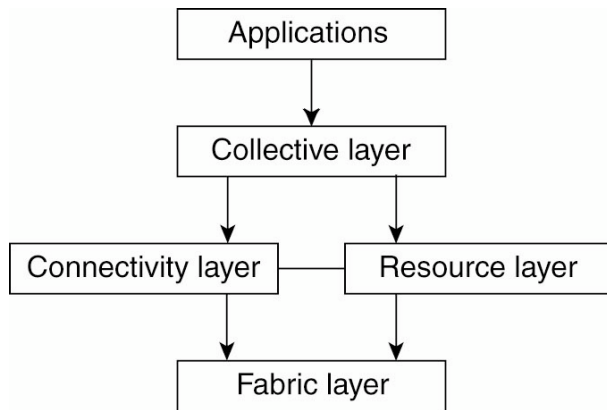
connectivity layer — communication protocols for grid transactions spanning over multiple resources, plus security protocols for authentication

collective layer — handling access to multiple resources—resource discovery, allocation, ...

application layer — applications operating in the virtual organisation



Architecture of a Grid Computing System II



[Tanenbaum and van Steen, 2007]

Architecture of a Grid Computing System III

Grid middleware layer

- the *core* of a grid middleware layer is represented by *connectivity*, *resource*, and *collective* layers
- altogether, they provide *uniform access* to otherwise dispersed *resources*



Next in Line...

- 1 Distributed Computing Systems
- 2 Distributed Information Systems**
- 3 Distributed Pervasive Systems



Distributed Information Systems

Origin

- Many separate networked applications to be integrated
- Structural problems of interoperability

Sorts

- Several non-interoperating servers shared by a number of clients: distributed queries, distributed transactions
- Transaction Processing Systems
- Several sophisticated applications – not only databases, but also processing components – requiring to directly communicate with each other
- Enterprise Application Integration (EAI)

Transaction Processing Systems

Distributed transactions for distributed databases

- operations on databases are usually performed in terms of **transactions**
- when databases are distributed, transactions should be *distributed*
- *special primitives* from the distributed system or the runtime system

ACID properties

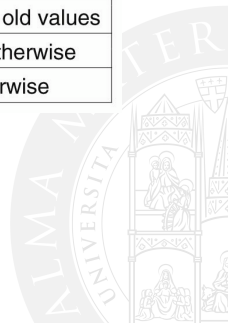
- atomic** steps of a transaction occurs *invisibly* to the outside world
- consistent** the transaction does *not violate* system *invariants*
- isolated** *concurrent* transactions do *not interfere* with each other
- durable** once a transaction *commits*, its effects are *permanent*



Example Primitives for Transactions

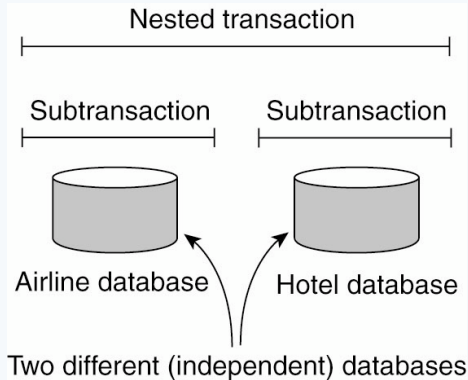
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

[Tanenbaum and van Steen, 2007]



Nested Transactions

A nested transaction is made of a number of subtransactions



[Tanenbaum and van Steen, 2007]

Nesting in transactions could be arbitrarily deep

The Problem with Nested Transactions I

Durability of nested and sub-transactions

- a *whole* nested transaction should exhibit ACID properties
 - so, if a subtransaction fails, all subtransactions till there should be *undone*, even though they already committed
 - the effects of subtransactions could not be really durable if the whole transaction does not succeed
- *durability* here refers to the *top-level* transaction



The Problem with Nested Transactions II

Private copy of the world as a solution

- all transactions are performed over a copy of the data, so subtransactions could keep ACIDity in the *local world*
 - the effect of a *successful* nested transaction would be *propagated* only *after it succeeds*
- in case, the *copy* of the world transformed *becomes* the *world*
- in any case, transactions are ACID



Nested Transactions in Distributed Information Systems

Nested transactions are a natural way for distributing transactions

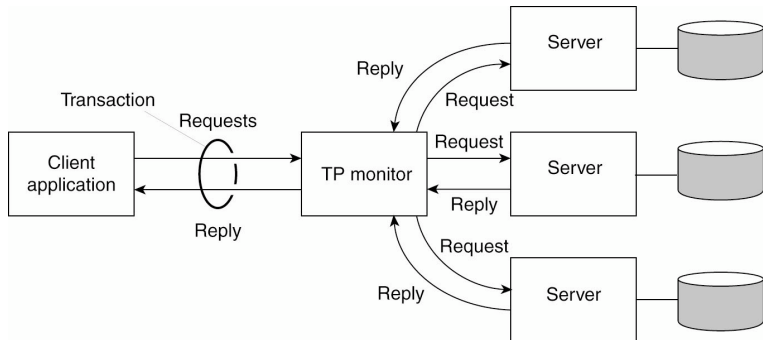
- “leaf” subtransactions are usual transactions over single servers
- distributed transactions are nested transactions

An early solution: TP monitor

- transaction processing monitor (or, *TP monitor*)
- to allow applications to access *multiple* DB servers
- with a *transactional semantics*



TP Monitor



[Tanenbaum and van Steen, 2007]

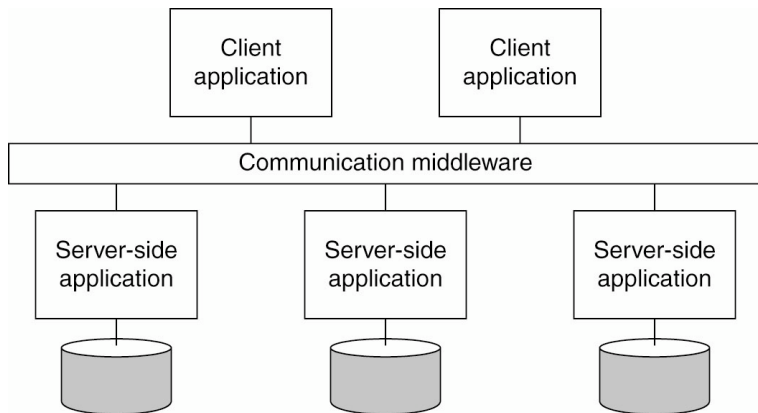
Enterprise Application Integration

It is not only a matter of accessing distributed databases

- integration should happen at the application level, too
- beyond data integration, process integration
- application should interact and communicate meaningfully with each other



Middleware as a Communication Facilitator



[Tanenbaum and van Steen, 2007]

Sorts of Communication Middleware

Different middleware supports different sorts of communication

RPC Remote Procedure Call

RMI Remote Method Invocation

MOM Message-Oriented Middleware

Publish & Subscribe



Next in Line...

- 1 Distributed Computing Systems
- 2 Distributed Information Systems
- 3 Distributed Pervasive Systems**



Distributed Systems with Instability

What happens when instability is the default condition?

- ... like, with mobile devices with batteries and sporadic network connection?
- ... like, in modern *distributed pervasive systems*?

Main features

- a **distributed pervasive system** is *part of our surroundings*
- a distributed pervasive system generally *lacks of a human administrative control*

Requirements for Pervasive Systems

Three requirements [Grimm et al., 2004]

- embrace contextual changes
- encourage ad hoc composition
- recognise sharing as the default

Remarks

- a device must be continually *aware* of the fact that its *environment* may change at any time
- many devices in pervasive system will be *used in different ways* by different users
- devices generally join the system in order to access (provide) information: *information* should then be easy to read, store, manage, and *share*

Home Systems

Systems built around home networks

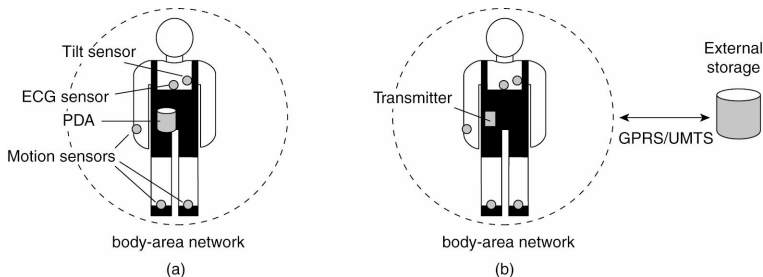
- no way to ask people to act as a competent network / system administrator
- home systems should be self-configuring and self-maintaining in essence

Systems built around personal information

- *huge amount* of heterogeneous personal *information* to be managed,
- coming from *heterogeneous sources* from inside and outside the home system

Health Care Systems

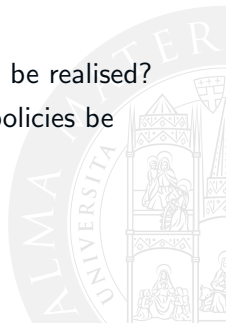
- Personal systems built around a Body Area Network
- Possibly, minimising impact on the person—like, preventing free motion



[Tanenbaum and van Steen, 2007]

Health Care Systems: Questions to be Addressed

- where and how should monitored *data* be *stored*?
- how can we prevent *loss* of crucial *data*?
- what infrastructure is needed to generate and propagate *alerts*?
- how can physicians provide online *feedback*?
- how can extreme *robustness* of the monitoring system be realised?
- what are the *security* issues and how can the proper policies be enforced?



Sensor Networks

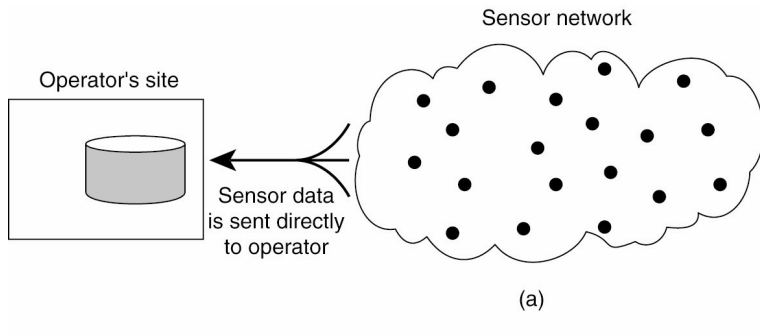
An enabling technology for pervasive systems

- clouds of *spatially-distributed sensors*—from tens to thousands of nodes with a sensing device
- acquiring, processing and transmitting *environmental information*

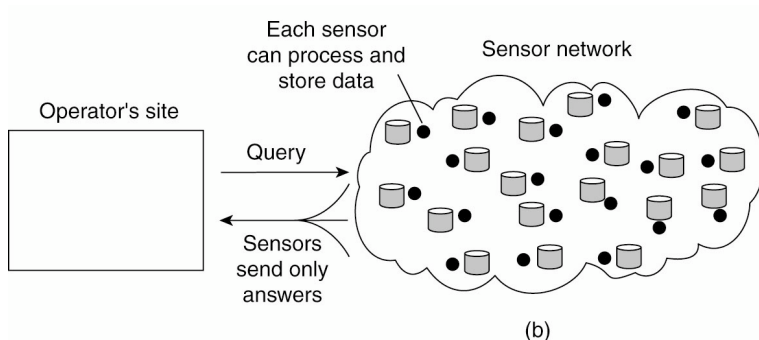
A possible view: distributed databases

- distributed sources of information
 - that can possibly be queried along time
 - two possible extremes: either sensors just send information in without cooperating, or they do all the computation and return the results
- ↑ both solutions are bad ones, since they require either too much network consumption or too much node power consumption

Architecture of Sensor Networks: Two Extremes I



Architecture of Sensor Networks: Two Extremes II



[Tanenbaum and van Steen, 2007]

Sensor Networks: A Solution

In-network data-processing

- setting up a tree network among sensors
- passing queries through the sensor tree
- aggregating the results at the different levels of the tree

Questions to be addressed

- how do we (dynamically) set up an efficient tree in a sensor network?
- how does aggregation of results take place? Can it be controlled?
- what happens when network links fail?

Summing Up

Diverse sorts of distributed systems exist

- depending on both the *environment* where they are developed, the *goals* they have to achieve, and the level of the available *technologies*
- different *models*, *methodologies*, and *technologies* are to be used to design and develop different sorts of distributed systems



References



Foster, I., Kesselman, C., and Tuecke, S. (2001).

[The anatomy of the grid: Enabling scalable virtual organizations.](#)

International Journal of High Performance Computing Applications, 15(3):200–222.



Grimm, R., Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., and Wetherall, D. (2004).

[System support for pervasive applications.](#)

ACM Transactions on Computer Systems, 22(4):421–486.



Tanenbaum, A. S. and van Steen, M. (2007).

[Distributed Systems. Principles and Paradigms.](#)

Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.



Sorts of Distributed Systems

Distributed Systems

Sistemi Distribuiti

Andrea Omicini

`andrea.omicini@unibo.it`

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna a Cesena

Academic Year 2016/2017

