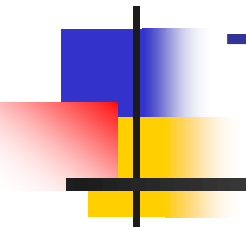


Distributed Systems II TDA297(CTH), DIT290 (GU)

LP3 2012
7.5 hec
http://www.cse.chalmers.se/edu/course/TDA297

The Teachers

- Instructor: Philippas Tsigas
  - E-mail: tsigas@chalmers.se
  - Phone: +46-31-772 5409
  - Office: 5106
  - Office hours: By appointment
- Assistants:
  - Farnaz Moradi – moradi@chalmers.se
  - Ioannis Nikolakopoulos – ioaniko@chalmers.se
  - Office hours: To be announced

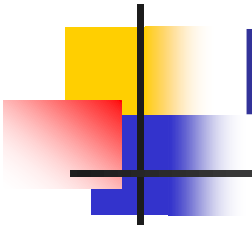
The Teachers

- Farnaz Moradi
  moradi@chalmers.se
  Phone: 772 1045
  Room: 5111
- Ioannis Nikolakopoulos
  ioaniko@chalmers.se
  Phone: 772 5720
  Room: 5107

- Office hours: To be announced

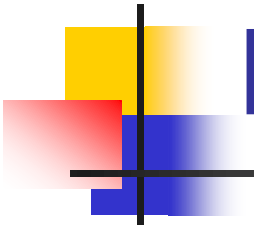
Course Representatives

Johansson	William	wiljoh@student.chalmers.se
Nowdehi	Nasser	nowdehi@student.chalmers.se
Svensson	Martin	martsven@student.chalmers.se
Söderberg	Mathias	soderbem@student.chalmers.se
Walulya	Ivan	walulya@student.chalmers.se



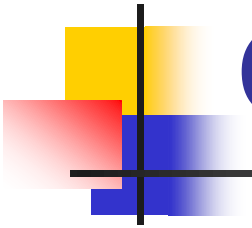
Reading

- 4th Edition of the book: "Distributed Systems: Concepts and Design"
  - written by: George Coulouris, Jean Dollimore and Tim Kindberg
  - published by Addison-Wesley, ISBN 0-321-26354-5
- Some extra material: Distributed Algorithms (Notes) + papers.



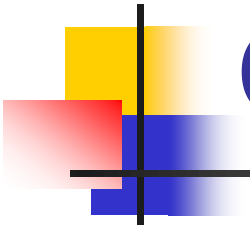
Description

Distributed systems are popular and powerful computing paradigms. Their importance increases as networked computers become more common than freestanding ones, especially since many different types of computers can be found in networks. In this course we will see the points of inherent difference and strength of distributed systems compared with sequential or strongly-coupled systems; consequently, we will also study the issues and problems that have to be addressed and solved efficiently for these differences to be taken advantage of, so that the system retains its strength and high potential.



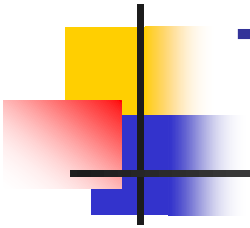
Goals

- 1.Replication; The advantages and costs of replicating data:
  - Potential improvement in response times and reliability
  - Extra communication costs involved in keeping data consistent.
- 2.Fault-tolerant Agreement in Distributed Systems:
  - (a very special and significant problem, since it is a key issue in most synchronisation and coordination problems in distributed systems)
  - Study of the basic impossibility results and discuss their implications
  - Proceed with solutions and protocols for systems with certain strengths and design structures.



Goals cont.

- 3. Resource Allocation.
- 4. Distributed algorithms: How to design and analyse distributed algorithms?
- 5. Sensor Networks.
- 6. Denial of Service Attacks.

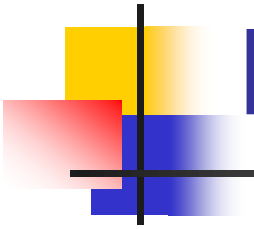


This is an advanced course

- No list of things to learn
  - The book and assignments cover most exam material
- Find out what you need
  - Lectures
  - The book
  - Internet
- Lectures and assignments in parallel
  - Don’t wait for the lectures before you start with the assignments

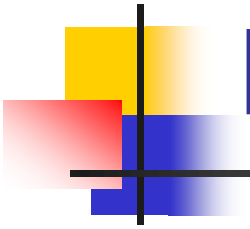
Approximate Course Schedule

Week (Chalmers week/LV)	Monday	Wednesday	Friday
3 (1) (Jan 17-21)	**Lecture 1:** Introduction	**Lecture 2:** Broadcasting	**Assignment Lecture 1:** Introduction to Lab 1
4 (2) (Jan 24-Jan 28)	**Lecture 3:** Broadcasting & Replication	**Lecture 4:** Replication	**Lecture 5:** Replication & Quorum Consensus
5 (3) (Mar 31-Feb 4)	**Assignment Lecture 2:** Introduction to Lab 2 + Questions and answers for all labs.	No Lecture – Work on assignments	**Lecture 6:** Distributed denial of service attacks
6 (4) (Feb 7-11)	**Lecture 7:** Distributed Transactions & Concurrency Control in Distributed Transactions Deadline for lab 1 at 23:59	CHARM – No Lecture	**Assignment Lecture 3:** Questions and answers for all labs.
7 (5) (Feb 14-18)	**Lecture 8:** Atomic Commit protocols & Byzantine General	**Lecture 9:** Distributed Algorithms	**Assignment Lecture 4:** Sensor Networks + Introduction to Lab 3 + Questions and answers for all labs.
8 (6) (Feb 21-25)	**Lecture 10:** Grouping and routing in sensor networks Deadline for lab 2 at 23:59	**Assignment Lecture 5:** Questions and answers for all labs.	**Lecture 11:** Mutual Exclusion and Resource Allocation. Dining Philosophers
9 (7) (Feb 28-Mar 4)	**Lecture 12:** Generalization of the Dining Philosophers	**Lecture 13:** Drinking Philosophers and Efficient Resource Allocation	No Lecture Deadline for lab 3 at 23:59
10 (8) (Mar 7-11)	**Assignment presentations:** Presentations and Demos of the 3rd lab.	**Lecture 14:** Efficient Resource Allocation continued	**Lecture 15:** Closing
	EXAM: Monday the 14th of March 2011 in the V building 14:00-18:00		



Examination

- 1st Lab due: arround 10th of February
- 2nd Lab due: arround 17th of February
- 3rd Lab due: arround 1st of March
- Final examination: 05th of March



Resources

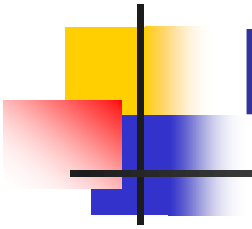
Full support page for the Coulouris' book:

- http://www.cdk4.net/
- http://www.cdk5.net/

Slides:

- At homepage after lecture
- Last years slides available, use them as a reference point before the lecture: http://www.cse.chalmers.se/edu/year/2010/course/TDA297/

Resources cont.



1st Assignment:

- Distributed bulletin board
- http://www.cse.chalmers.se/edu/course/TDA297/labs/lab1.html

Resources cont(2).

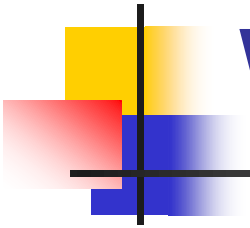
2nd Assignment:

- Reliable and ordered multicast
- http://www.cse.chalmers.se/edu/course/TDA297/labs/lab2.html

3nd Assignment:

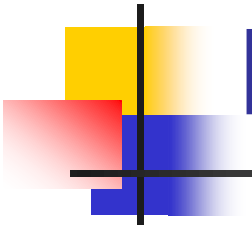
- Routing in Sensor Networks
- http://www.cse.chalmers.se/edu/course/TDA297/labs/lab3.html


LEACH



Why Distributed Systems?

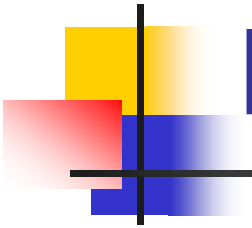
Suggestions?



Because they are there

Distributed Applications

- Automated Banking Systems
- Tracking Roaming Cellular Telephones
- Air-Traffic Control Systems
- Fly-by-wire Systems
- The World-Wide- Web

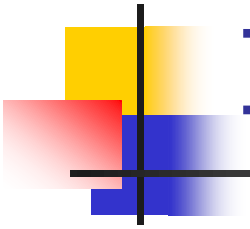


Because they have many +

Network vs. Centralised Systems

- Cycles always available
- Incremental growth
- Independent failure
- Heterogeneity
- Increased Autonomy
  - purchasing
  - management
  - software

Ingredients of D.S.

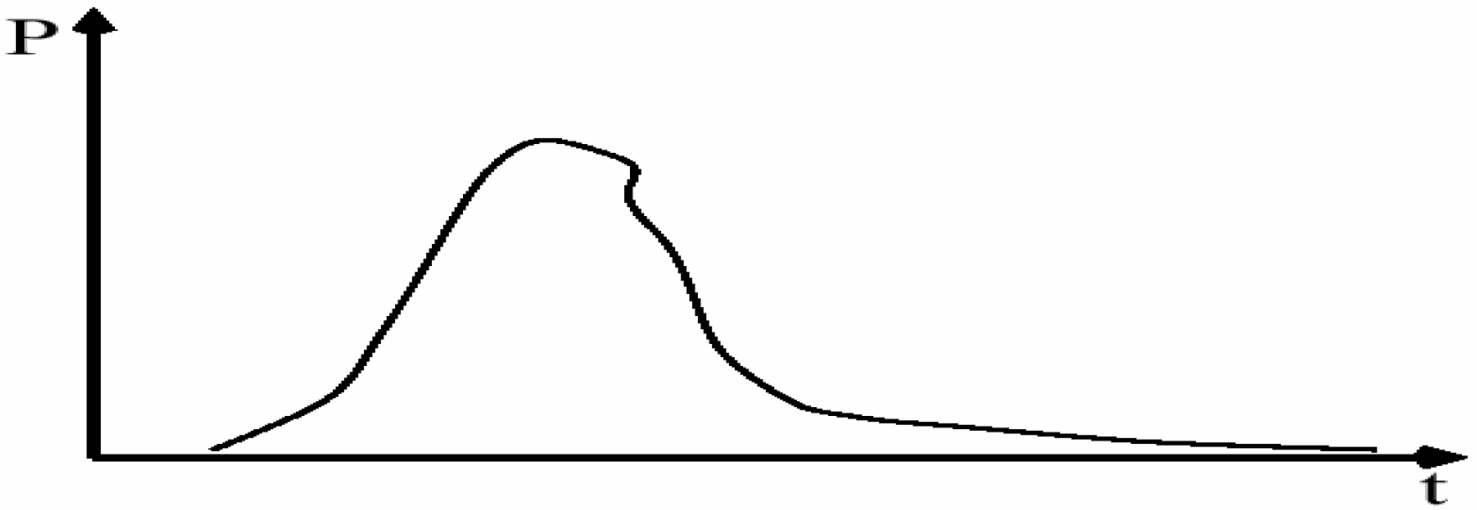


- Multiple computers
- Interconnections

Shared states

Asynchronous Systems

- Communication is reliable but incurs potentially unbounded delays



Distributed Computation

A distributed computation is a *single execution* of a *distributed program* by a collection of *processes*. Each sequential process generates a sequence of *events* that are either *internal events*, or *communication events*

The *local history* of process p_i during a computation is a (possibly infinite) sequence of events $h = e_i^1, e_i^2, \ldots$

The *global history* of a computation is a set $H = \bigcup_{i=1}^{n} h_i$

State of the computation

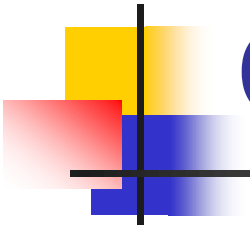
- Imagine stopping a distributed computation by **stop**ping **all** of its **processes simultaneously**.
- The combined **states of** each of the **processes, plus** the contents of the **messages in transit** between processes will then tell us the exact **global state of the computation**.

State of the computation

- The problem with our **asynchronous system** is that there is **no such thing as simultaneity**.
- The concept of a system state is still used in analysis

The local state of process p_i after executing event e_i^k is denoted σ_i^k. The initial local state of p_i is σ_i^0.

A *global state* is a collection of local states $\Sigma = \bigcup_{i=1}^{n} \sigma_i$.



Cause and Effect

- No simultaneity in an asynchronous system

  => we need something else in order to study the events in such a system.
- We do have the notion of cause and effect, however.
- If one event e_i caused another event e_j to happen, then e_i and e_j could never have happened simultaneously:

  e_i happened before e_j

Causal Orders

We define a binary relation $\rightarrow$ over events, such that

1. If $e_i^k, e_i^l \in h_i$ and $k < l$, then $e_i^k \rightarrow e_i^l$
2. If $e_i = \text{send}(m)$ and $e_j = \text{receive}(m)$, then $e_i \rightarrow e_j$
3. If $e \rightarrow e'$ and $e' \rightarrow e''$, then $e \rightarrow e''$

When $e \rightarrow e'$, we say e *causally precedes* e' or e *happened before* e'.

We define *concurrent* as $e \parallel e' \equiv \neg(e \rightarrow e' \vee e' \rightarrow e)$

Space Time Diagram

A distributed computation is a *partially ordered set* (poset) $y = (H, \rightarrow)$.

A distributed computation can be depicted in a *space-time diagram*: