Introduction to Distributed Systems

Introduction

Unil | HEC | dop lab
Benoît Garbinato
distributed object programming lab

Distributed systems (1)

"A distributed system is one that stops you from getting any work done when a machine you've never even heard of crashes."

L. Lamport, quoted by S. Müllender in Distributed Systems. 2nd edition. Addison-Wesley, 1993.

Distributed systems (2)

"As long as there were no ~~machines~~ networks, programming was no problem at all; when we had a few weak ~~computers~~ networks, programming became a mild problem and now that we have gigantic ~~computers~~ networks, programming has become an equally gigantic problem. In this sense the electronic industry has not solved a single problem, it has only created them - it has created the problem of using its products."

distributed

distributed

distributed

Edgster Dijkstra, The Humble Programmer.
Communication of the ACM, vol. 15, no. 10.
October 1972. Turing Award Lecture.

Distributed systems (3)

"A distributed system is a collection of autonomous computers linked by a network, with software designed to produce an integrated computing facility."

In Distributed Systems: Concept and Design.
2nd edition. Addison-Wesley, 1994.

Historical background

- Hardware became continuously cheaper
- Cheap and fast networks emerged
- The example of Unix:

  1969 K. Thompson & D. Ritchie develop Unix as a multi-users system on PDP-7

  1979 B. Joy enhances Unix with interprocess communication facilities (BSD Unix)

  1980's Sun Microsystems used BSD Unix as operating systems for its workstations

Approach of this course (1)

- This course teaches distributed systems from both a practical and a theoretical perspective

"In theory, there is not difference between theory & practice. In practice, there is."

- The practitioner needs the theoretical perspective to understand the implicit assumptions hidden in the technologies, and their consequences
- The theoretician needs the practical perspective to validate that theoretical models, problems & solutions work in accordance to existing technologies

Approach of this course (2)

To achieve this, we will approach distributed systems through four complementary views:

- The model view
- The interaction view
- The architecture view
- The algorithm view

The model view

- What distributed entities?
  E.g., processes, objects, threads, etc.
- What time assumptions?
  E.g., synchronous, asynchronous, etc.
- What failure assumption?
  E.g., crash-stop, malicious, etc.

The interaction view

- What interaction paradigm?
  E.g., message passing, shared memory, etc.
- What reliability guarantees?
  E.g., best-effort, reliable, secure, etc.

The architecture view

- What level of decentralization?
  E.g., client/server, multi-tier, etc.
- What level of separation of concerns?
  E.g., library-based, container-based, etc.

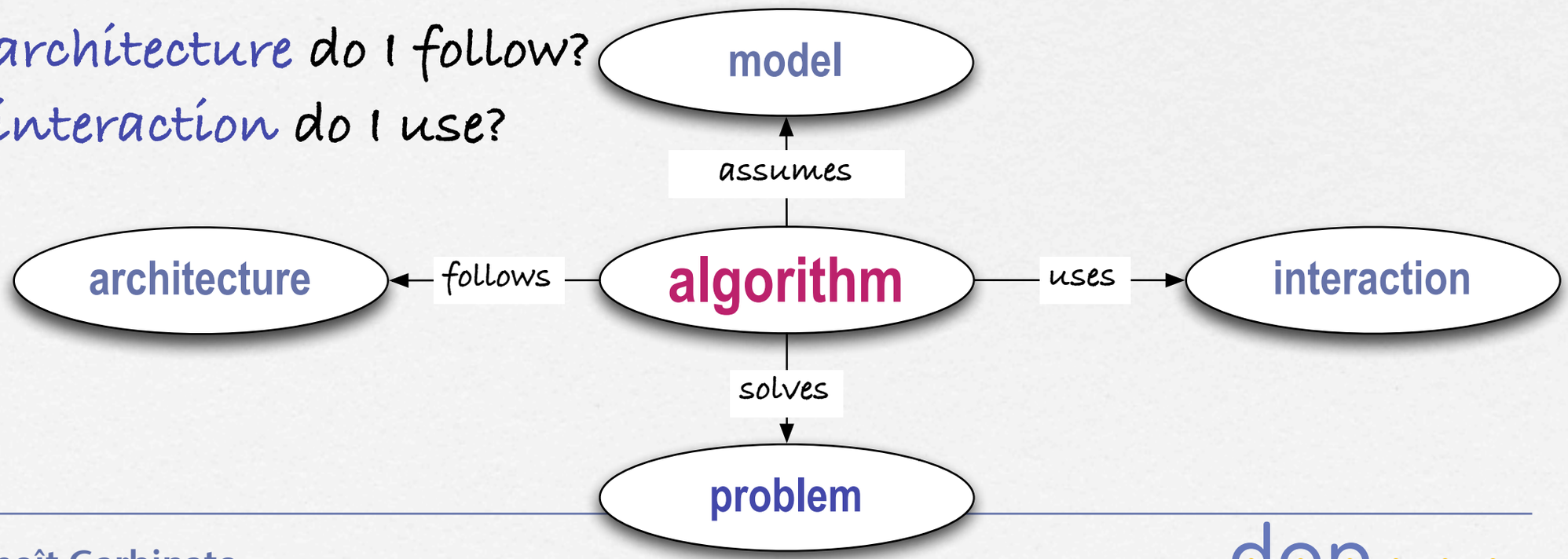
The algorithm view

- What problem?
  E.g., internet payment, consensus, etc.
- What algorithm?
  E.g., two phase commit, sliding window, etc.
- What complexity and what performance?
  E.g., NP-complete, polynomial, etc.

The big picture

When implementing a distributed program, you will always end up writing some algorithm. In doing so, you will have to answer the following questions:

- ☐ What problem am I trying to solve?
- ☐ What model do I assume?
- ☐ What architecture do I follow?
- ☐ What interaction do I use?



Content overview

- Remote method invocation
- Basics of distributed algorithms
- Concurrent & network programming
- Mobile distributed programming

Technologies we will use

- Java programming platform
- Objective-C + iOS software platform
- Internet protocols (TCP, UDP)

Organization

- Lectures + exercises + practical project
- Evaluation :
  - Project (P) – group project (compulsory)
  - Final exam (E) – individual exam (compulsory)

If $E \geq 3$: Final grade $= 0.5 \times P + 0.5 \times E$

If $E < 3$: Final grade $= E$

- For the project, the mark of each member of a group might vary, based on participation

Exercises & the project

- Exercises should help you get started with individual technologies presented
- The project should allow you to understand how technologies can be combined to devise a complete solution... and have fun!

some business logic | distribution technologies | other business logic

The project

- The subject of the project is free but must have a distributed nature and be based on the concepts & tools presented in the lecture and exercise sessions
- Projects are done in groups (membership may slightly vary between groups this is taken into account when grading)

Timetable

	13:15 - 14:00	14:15 - 15:00	15:15 - 16:00	16:15 - 17:00
Sep 22, 2016	Introduction	Remote Method Invocation		Discover Lab Tools
Sep 29, 2016	Exercises [Java RMI]			
Oct 6, 2016	Concurrent Programming			Exercises
Oct 13, 2016	Network Programming			
Oct 20, 2016	Basic Programming in Objective-C		Exercises	Project kickoff
Oct 27, 2016	Project specification			
Nov 3, 2016	Intermediate Presentation | Specification Validation		Project Implementation	
Nov 10, 2016	Network Programming in Objective-C		Exercises	Project Implementation
Nov 17, 2016	Thematic Week			
Nov 24, 2016 Dec 1, 2016	Distributed Algorithms		Project Implementation	
Dec 8, 2016 Dec 15, 2016	Project Q&A			
Dec 22, 2016	Final Presentation | Project Demo & Assessment			

Further information

- http://doplab.unil.ch/ids
- arielle.moro@unil.ch
- vaibhav.kulkarni@unil.ch
- benoit.garbinato@unil.ch
- Interesting book:

  Distributed Systems - Concepts and Design, 4th Edition, J. Dollimore, T. Kindberg, G. Coulouris, Addison Wesley / Pearson Education, 2005.