Distributed Systems and Algorithms

Martin Quinson <martin.quinson@loria.fr>
Abdelkader Lahmadi <abdelkader.lahmadi@loria.fr>
Olivier Festor <olivier.festor@inria.fr>

LORIA – INRIA Nancy Grand Est

2018-2019

(compiled on: September 26, 2019)

Introduction

Course Goals

- Introduce existing distributed systems, from a theoretical point of view
  - Basic concepts
  - Main issues, problems and solutions

Motivations

- Distributed Systems more and more mainstream
- Interesting algorithmic issues

Administrativae

Contents

- Quick recap of distributed algorithmic and Internet
- Present several innovative distributed systems

Evaluation: test on desk *(partiel)*

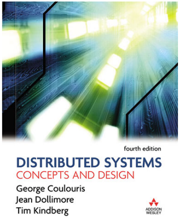
- What: quiz about the lectures
  - *Know* the algorithms introduced in lectures
  - Be able to *recognize* principle of classical algorithm designs
  - Be able to *discuss* the validity of an approach to a problem

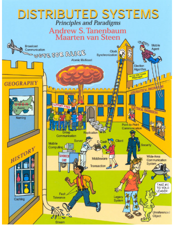
References: Courses on Internet

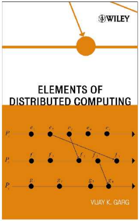
- Algorithmique et techniques de base des systèmes répartis (S. Krakowiak)
  Foundations of distributed systems (in French).
  `http://sardes.inrialpes.fr/~krakowia/Enseignement/M2R-SL/SR/`
- Distributed Systems (Shenker, Stoica; University of California, Berkley)
  A bit of everything, emphasis on Brewer's conjecture.
  `http://inst.eecs.berkley.edu/~cs194`
- Peer-to-Peer Networks (Jussi Kangasharju)
  Peer-to-peer systems.
  `http://www.cs.helsinki.fi/u/jakangas/Teaching/p2p-08f.html`
- Advanced Operating Systems (Neeraj Mittal)
  Very good presentation of the theoretical foundations.
  `http://www.utdallas.edu/~neerajm/cs6378f09`
- Grid Computing WS 09/10 (E. Jessen, M. Gerndt)
  Grid and Cloud computing.
  `http://www.lrr.in.tum.de/~gerndt/home/Teaching/WS2009/GridComputing/GridComputing.htm`

References: Books

- Coulouris, Jean et Kindberg. Distributed Systems: Concepts and Design.
- Tannenbaum, Steen. Distributed Systems: Principles and Paradigms.
- V. K. Garg. Elements of Distributed Computing.
- Ralf Steinmetz, Klaus Wehrle (Eds): Peer-to-Peer Systems and Applications.
  `http://www.peer-to-peer.info/`







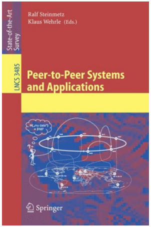


Table of Contents

1. Introduction to Distributed Systems
   - What is it? Examples.
2. Distributed Algorithmic
   - Time and state; Ordering events; Abstract Clocks; Classical Algorithms.
3. Mutual Exclusion and synchronisation
   - Introduction; classical algorithms and solutions to mutual exclusion problems.
4. Blockchains
   - Introduction; structure and components; consensus algorithms

Chapter 1

Introduction

- What is a Distributed System?
- Example of Distributed Systems
- Limit between Computers and Distributed Systems

What is a distributed system?

Definition

A distributed system is a collection of independent computers that appear to the users of the system as a single computer.

— *A. Tanenbaum.*

$\rightsquigarrow$ Set of elements (CPU, storage) interconnected by the network

- The set is more than the sum of its parts (elements do collaborate)
- Intuitive examples not from CS
  - Ant nest
  - Driving rules (cars share the road)

What is a distributed system?

Definition (optimistic)

A distributed system is a collection of independent computers that appear to the users of the system as a single computer.

— A. Tanenbaum.

$\leadsto$ Set of elements (CPU, storage) interconnected by the network

- The set is more than the sum of its parts (elements do collaborate)
- Intuitive examples not from CS
  - Ant nest
  - Driving rules (cars share the road)

Definition (pessimistic)

You know you have one when the crash of a computer you never heard of stops you from getting any work done. *— L. Lamport.*

- Interdepending behavior of elements
  - That's not that easy
  - Failures do happen and must be dealt with

Why would you distribute your computer system??

Application needs: you sometimes have to

- Collaborative work (between human beings, between corporate facilities)
- Distributed electronic devices ⇒ *Ubiquitous Computing* and *SensorNets*
- Application integration (multi-physics simulation) ⇒ *Grid Computing*

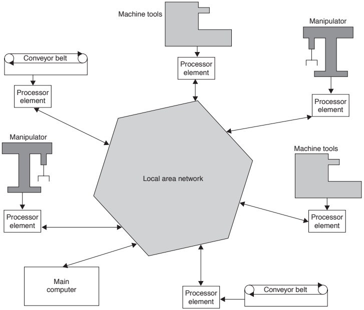
Technical possibility creates the need

- Cost effectiveness
  - A set of PC is less expensive than a big mainframe ⇒ *Cluster Computing*
  - Scale savings of mesocenter (wrt than several clusters) ⇒ *Cloud Computing*
- Generalized interconnections (TV, Internet, phone are converging)
  - Share storage resources ⇒ *Peer-to-Peer systems*
  - Share (otherwise unused) computational resources ⇒ *Volunteer Computing*

Example of Distributed Systems (1/3)

Manufacturing distributed system

- Embedded command and control systems
- Data acquisition processes



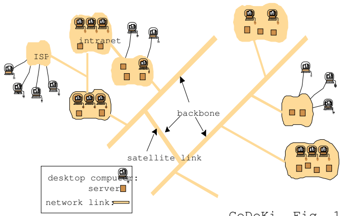
Aircraft

- Embedded distributed systems
- Integrated Modular Avionics : processing modules interconnected via ARNIC 629 bus

Example of Distributed Systems (2/3)

The Internet: the network of networks

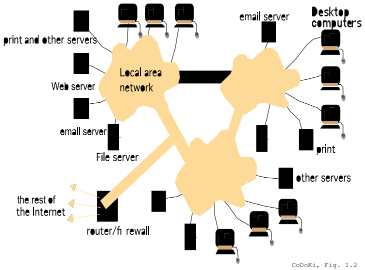
- Enormous (open ended)
- No single authority
  (mapping internet is a research agenda)
- Data, audio, video; Requests, push, streams.



CoDoKi, Fig. 1.1

Intranets

- A single authority
- Protected access
  (firewall, encrypted channels, total isolation)
- May be worldwide

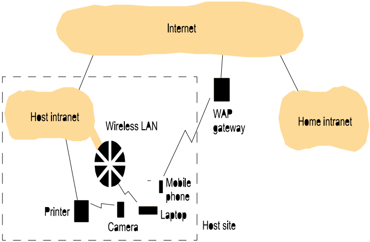


CoDoKi, Fig. 1.2

Example of Distributed Systems (3/3)

Mobile and Ubiquitous Computing

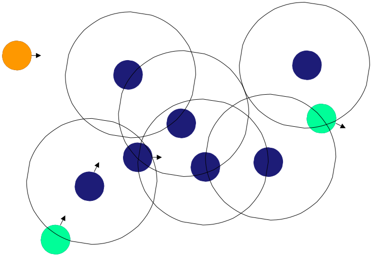
- Portable devices
  - laptops, notebook
  - handheld, wearable devices
  - devices embedded in appliances
- Mobile computing
- Connected to Internet through fixed infrastructure



(CoDoKi Fig 1.3)

Mobile Ad-hoc Networks (Manets)

- No fix infrastructure
  - wireless communication
  - multi-hop networking
  - long, non deterministic delays
  - ⇝ nodes part of infrastructure
- Nodes come and go

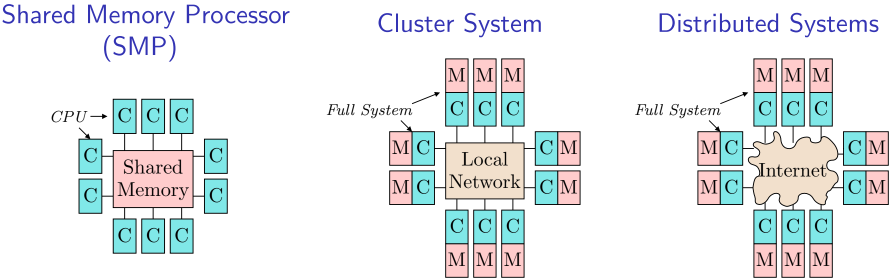


Limit between Computers and Distributed Systems

Why is this limit blurred?

- Motivation: endless need for power (modeling/game realism, server scalability)
- Past solution: Increase clock speed, more electronic gates
  (but reaching physical limits + speed linear vs. energy quadratic)
- Current trend: Multi-many (Multiply cores, processors and machines)

Multi-processors systems



- SMP communicate through shared memory
- Clusters and DS communicate through classical network

Distributed, Parallel or Concurrent??

Distributed Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow 0$

- Computation negligible wrt to communications
- Classical metric: amount of messages (as a function of amount of nodes)

Parallel Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \approx 1$

- Computation and Communication comparable
- Classical metric: makespan (time to completion of last processor)

Concurrent Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow \infty$

- Communication negligible wrt computation ($\textit{comm time} = 0 \Rightarrow$, multi-threading)
- Classical metric: speedup (how faster when using N cpus)

Distributed, Parallel or Concurrent??

Distributed Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow 0$

- Computation negligible wrt to communications
- Classical metric: amount of messages (as a function of amount of nodes)

Parallel Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \approx 1$

- Computation and Communication comparable
- Classical metric: makespan (time to completion of last processor)

Concurrent Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow \infty$

- Communication negligible wrt computation ($\textit{comm time} = 0 \Rightarrow$, multi-threading)
- Classical metric: speedup (how faster when using N cpus)

Focus of this course: distributed systems (some content applies to others)

Distributed, Parallel or Concurrent??

Distributed Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow 0$

- Computation negligible wrt to communications
- Classical metric: amount of messages (as a function of amount of nodes)
- Current research agenda: P2P, consistency (distributed DB)

Parallel Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \approx 1$

- Computation and Communication comparable
- Classical metric: makespan (time to completion of last processor)
- Current research agenda: Cluster & Grid & Cloud Computing, interoperability

Concurrent Algorithm: $\frac{\textit{computation time}}{\textit{communication time}} \rightsquigarrow \infty$

- Communication negligible wrt computation ($\textit{comm time} = 0 \Rightarrow$, multi-threading)
- Classical metric: speedup (how faster when using N cpus)
- Current research agenda: Lock-free, wait-free, correctness (model-checking)

Focus of this course: distributed systems (some content applies to others)

- Each domain constitutes a huge research area
- Current trend: intermixing, but strong historical heritage

What to expect from a distributed system?

Expected characteristics

- Scalability: deal with large amount of work
- Failure tolerance:
  - Deal with the failure of elements
  - Deal with message loss, or element performance degradation
- Security: Deal with malicious users (Privacy, Integrity, Deny-of-Services)
- Adaptability: deal with environment changes

Expected difficulties

- Absence of Global Clock: there is no common notion of time
- Absence of Shared Memory: no process has up-to-date global knowledge

- Failures (fail-stop or malicious): that *will* happen
- Delays (asynchronous): harder to detect failures
- Dynamism: global knowledge even harder to get

- Human brain is (somehow) sequential. Thinking distributed is harder.