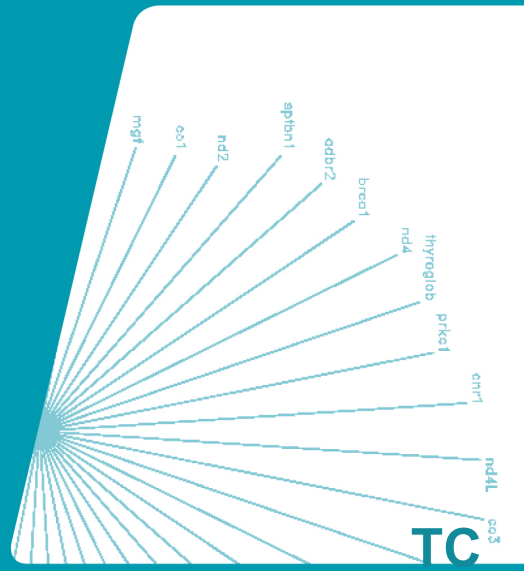


INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

Distributed Computing:

From Principles to Pervasive, Grid and Cloud Computing

membre de UNIVERSITÉ DE LYON

INSA LYON

Agenda

- Principles of Distributed Computing
  - Frédéric Le Mouël, (3*2h)
- Pervasive Computing
  - Frédéric Le Mouël, (4*2h)
- Cloud Computing
  - Julien Ponge, (4*2h)
- Grid Computing
  - Yves Caniou, (4*2h)

Successful lecture ?

- No mystery to succeed in research
  - Read books -> knowledge
  - Ask questions -> interactions
  - Criticize -> ideas
- Research evaluation
  - Exam, 2h
  - Ideas, method to answer a new problem

Who am I

- Frédéric Le Mouël
  - Associate Professor, INSA de Lyon, CITI Lab
  - Researcher, INRIA, Amazones Team
  - Topics: Middleware, OS, Java/OSGi, Mobile/ Pervasive Computing, Ambient Intelligence, Autonomic Computing

frederic.le-mouel@insa-lyon.fr @flemouel
http://www.le-mouel.net

DC Principles : Outline

- Introduction
  - What is Distributed Computing ?
  - Why distributing ?
  - What are the problems in distributing ?
- From theory …
  - Distributed Algorithms: Byzantine problem
    - Synchronizers, Logical clocks, etc.
    - Graph colouring, Mutual exclusion, Consensus, Self-stabilization, etc.
  - Complexity

DC Principles : Outline

- … to practice
  - Architectures
  - Message Passing
  - Message-Oriented Middleware (MOM)

What is Distributed Computing ?

« Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of **multiple autonomous computers** that communicate through a computer **network**. The computers interact with each other in order to achieve a **common goal**. A computer program that runs in a distributed system is called a distributed program, and **distributed programming** is the process of writing such programs »

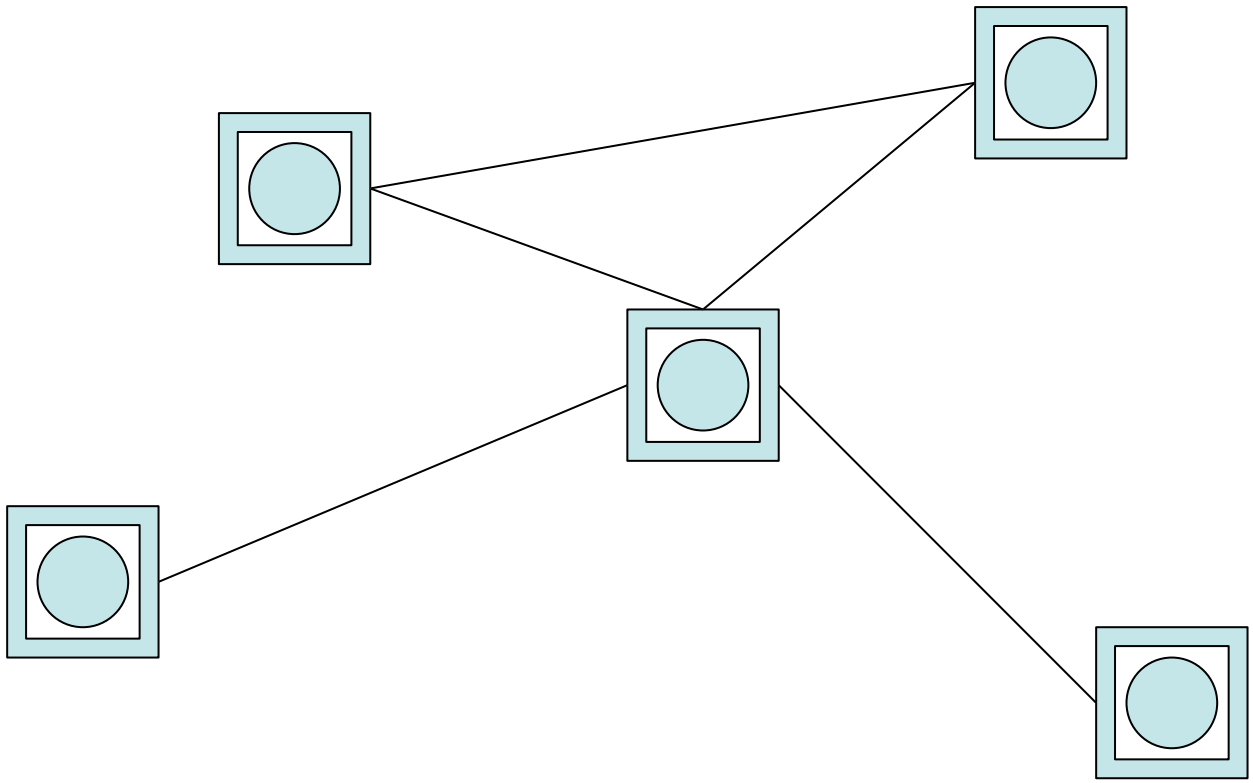
[wikipedia/DC] [Andrews 2000] [Ghosh 2007]

What is Distributed Computing ?

Autonomous Computers

Networks

Programs

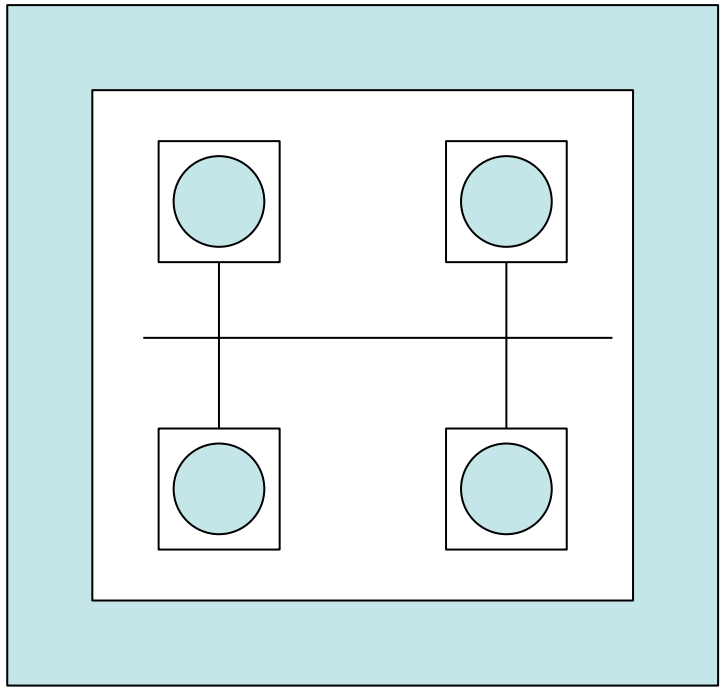


What is Distributed Computing ?

One Computer
Several Processors

Communication
Buses

Threads



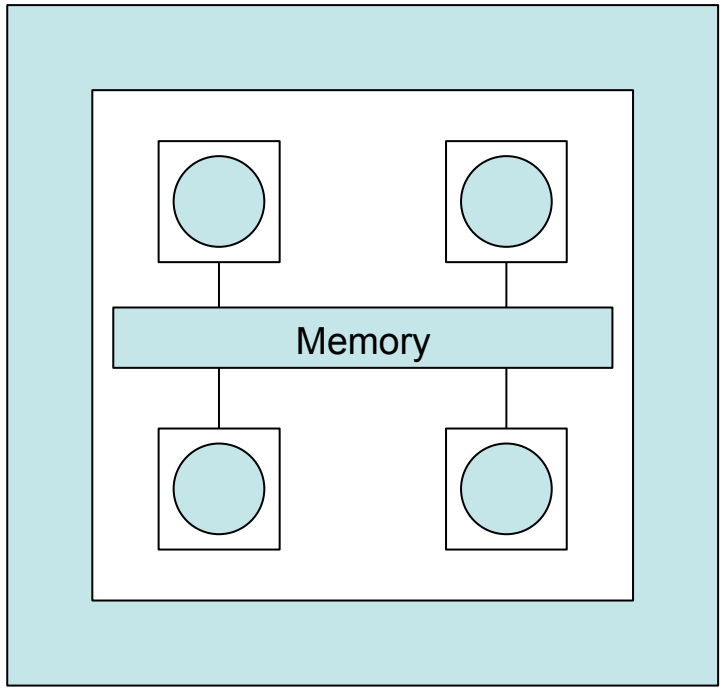
Parallel
What is ~~Distributed~~ Computing ?

One Computer
Several Processors

Communication
Buses

Threads

Shared Memory



Parallel computing ≈ tightly-coupled form of distributed computing

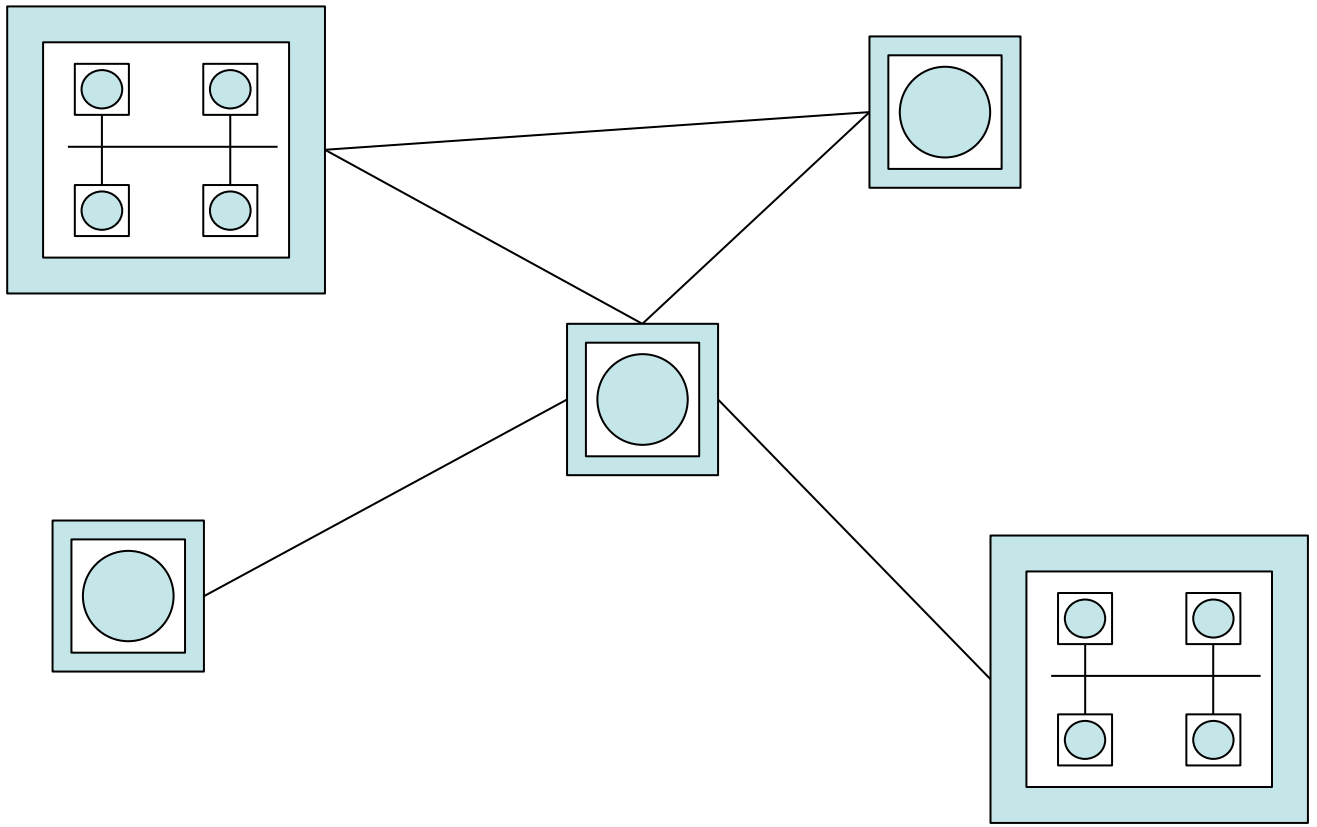
Distributed computing ≈ loosely-coupled form of parallel computing

What is Distributed Computing ?

Autonomous Computational Entities / Nodes (with their own memory)

Communication through Message Passing

Process



What is Distributed Computing ?

« Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of **multiple autonomous computers** that communicate through a computer **network**. The computers interact with each other in order to achieve a **common goal**. A computer program that runs in a distributed system is called a distributed program, and **distributed programming** is the process of writing such programs »

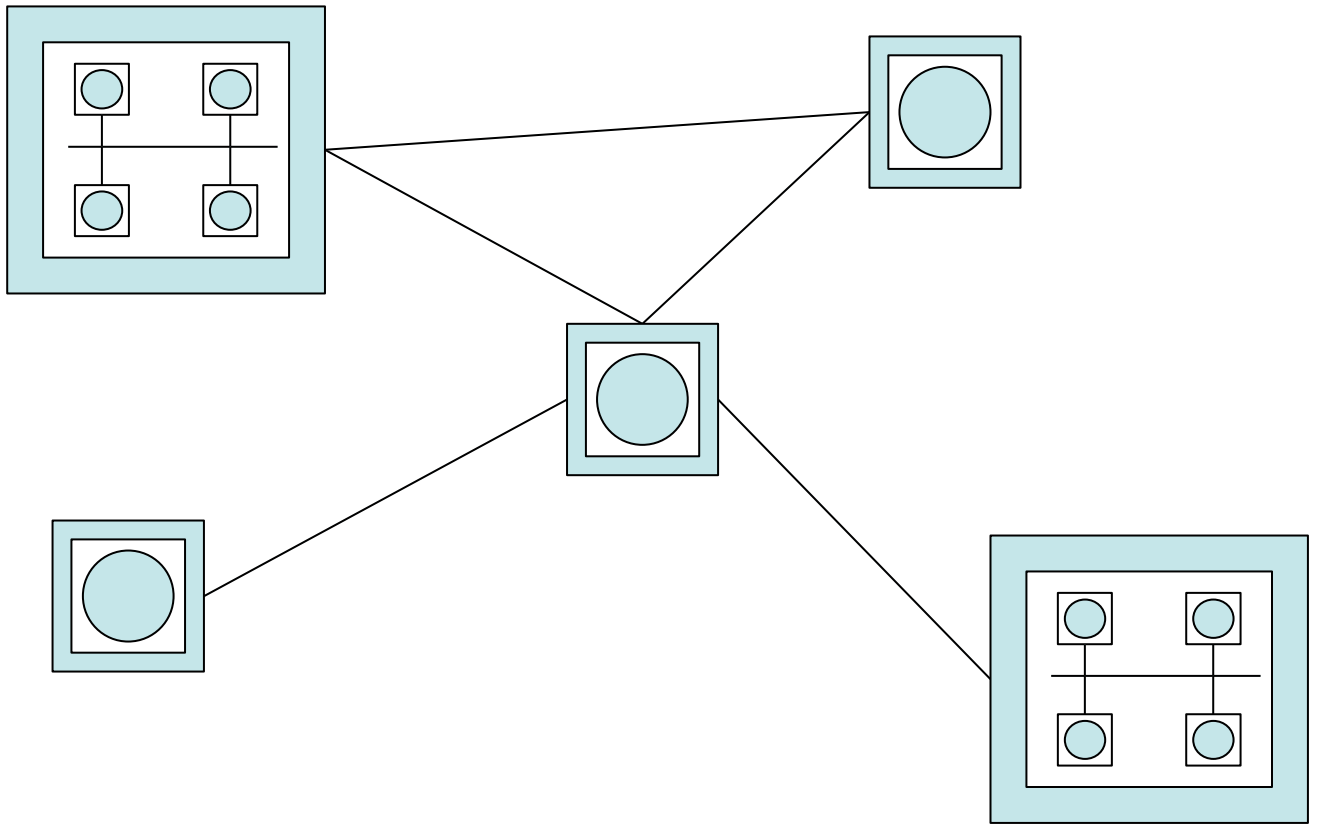
[wikipedia/DC] [Andrews 2000] [Ghosh 2007]

What is Distributed Computing ?

Common Goal ?

Process =
Memory zone split in

- Data segment
- Code segment
- Execution flow (stack, heap, etc.)

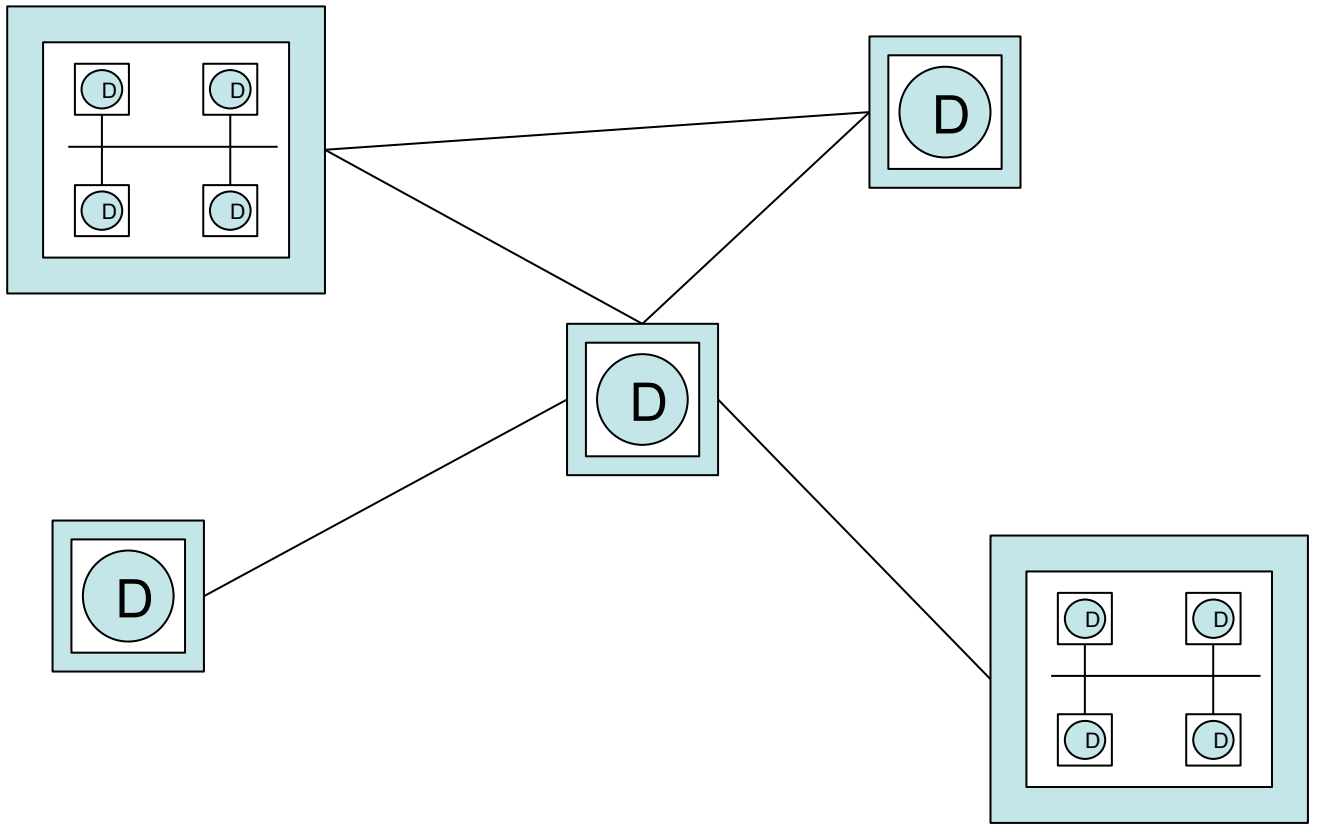


What is Distributed Computing ?

Goal = Physical Data Sharing

Process =
Memory zone split in

- **Data segment**
- Code segment
- Execution flow (stack, heap, etc.)

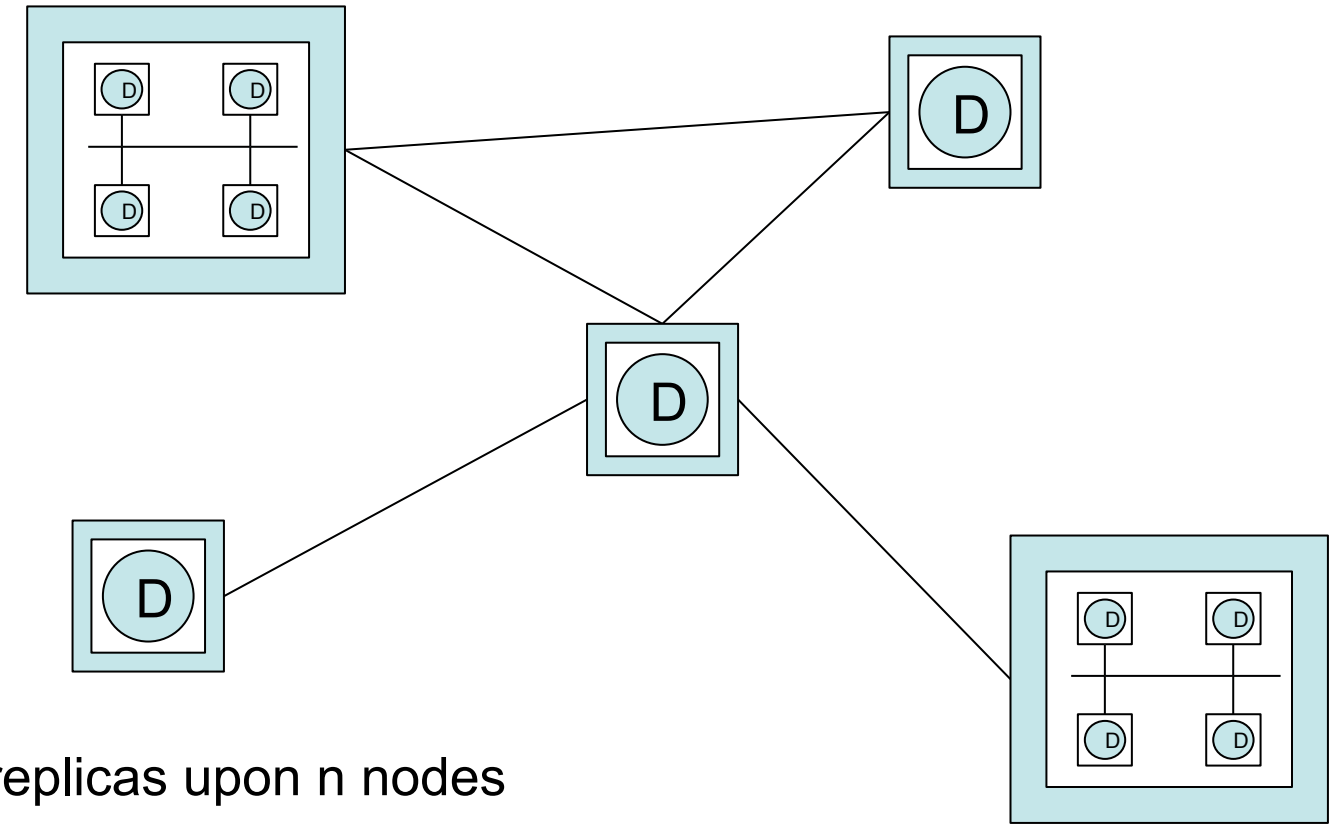


What is Distributed Computing ?

Goal = Physical Data Sharing

Process =
Memory zone split in

- **Data segment**
- Code segment
- Execution flow (stack, heap, etc.)



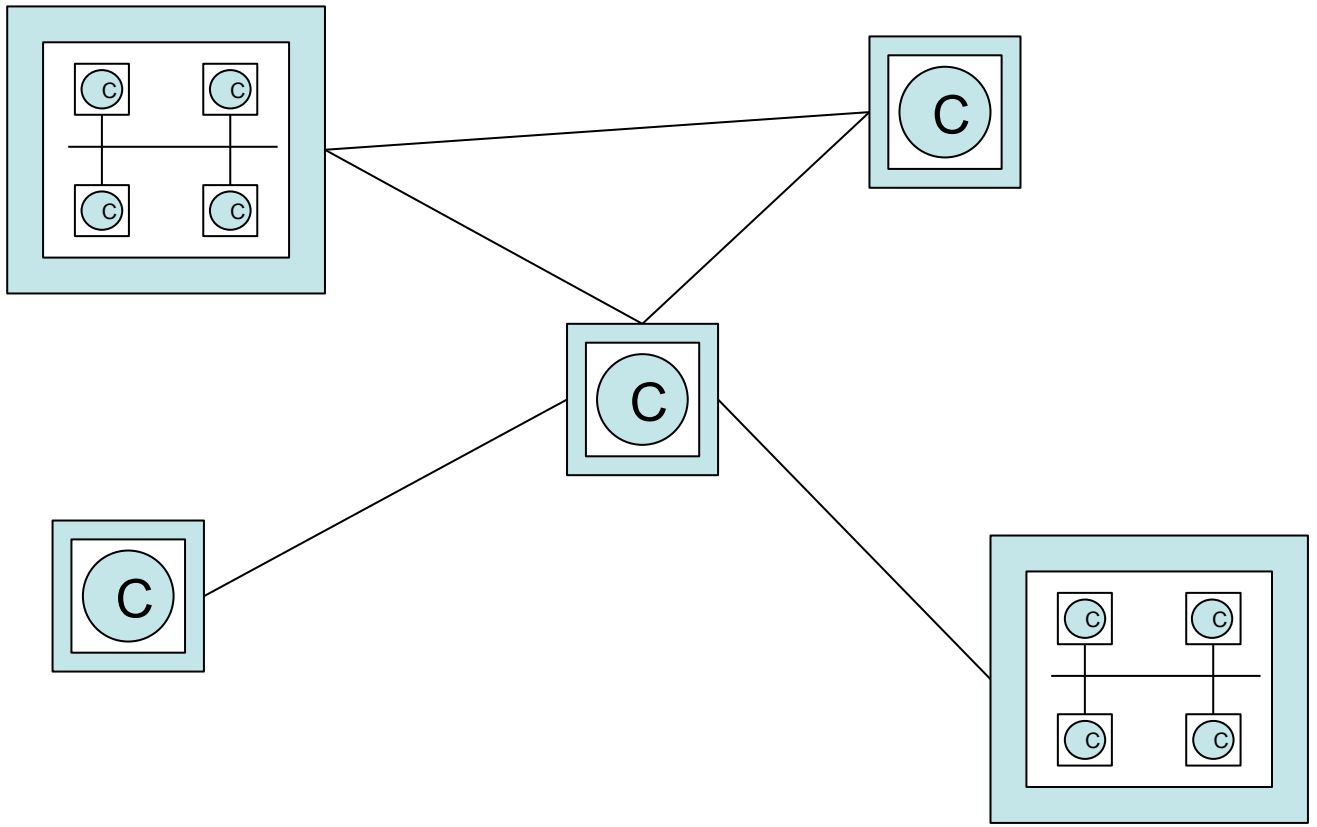
k replicas upon n nodes

What is Distributed Computing ?

Goal = Physical Code Sharing

Process =
Memory zone split in

- Data segment
- **Code segment**
- Execution flow (stack, heap, etc.)

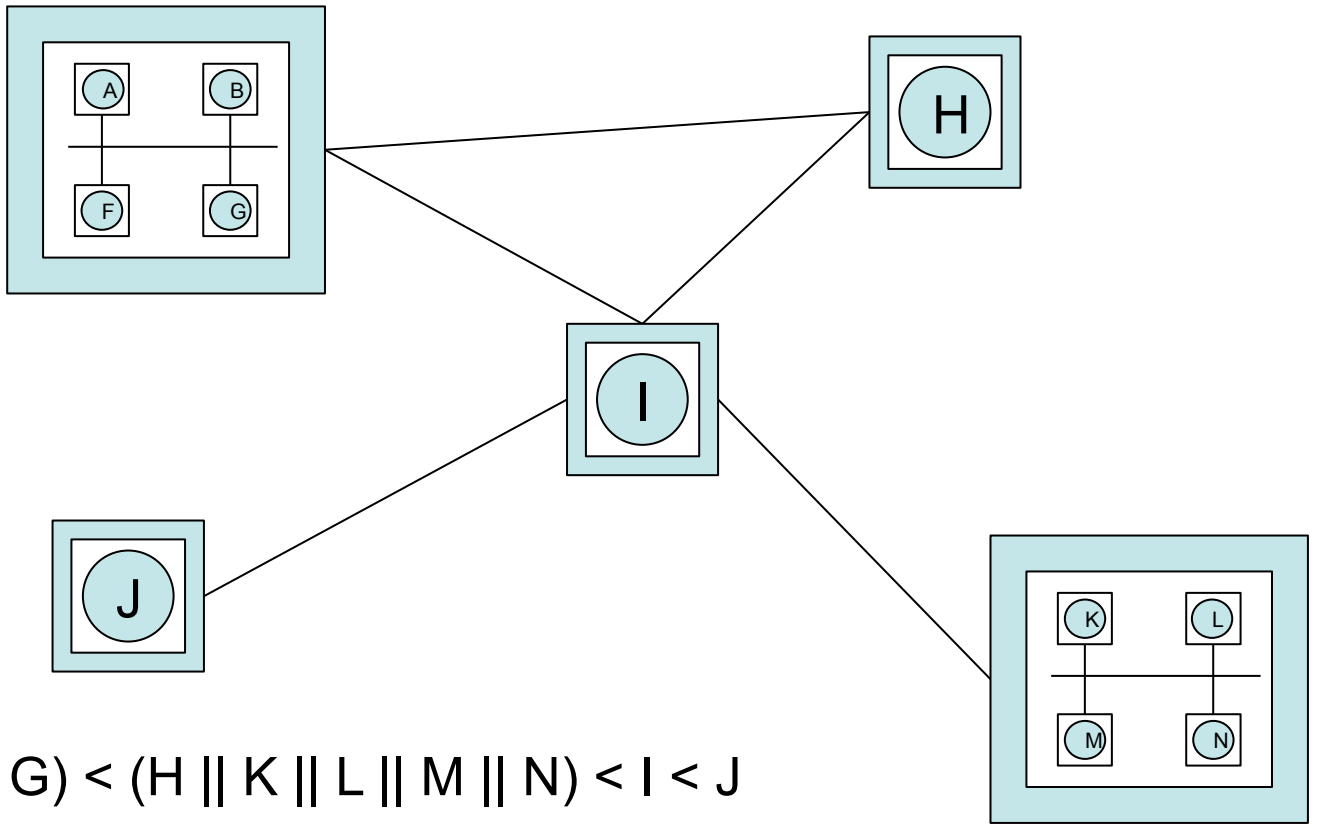


What is Distributed Computing ?

Goal = Temporal Synchronization of Computation

Process =
Memory zone split in

- Data segment
- Code segment
- **Execution flow (stack, heap, etc.)**



(A || B) < (F || G) < (H || K || L || M || N) < I < J

What is Distributed Computing ?

« Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of **multiple autonomous computers** that communicate through a computer **network**. The computers interact with each other in order to achieve a **common goal**. A computer program that runs in a distributed system is called a distributed program, and **distributed programming** is the process of writing such programs »

[wikipedia/DC] [Andrews 2000] [Ghosh 2007]

What is Distributed Programming ?

« Computer programming (often shortened to programming or coding) is the process of **designing, writing, testing, debugging, and maintaining** the source of computer programs. This source code is written in one or more programming languages. The purpose of programming is to create a program that performs specific operations or exhibits a certain desired behaviour. »

[wikipedia/CP]

Section 2: From theory …

Section 3: … to practice

Distributed Programming:

Designing = Distributed Algorithms

Writing = Programming Languages/Frameworks for Distribution

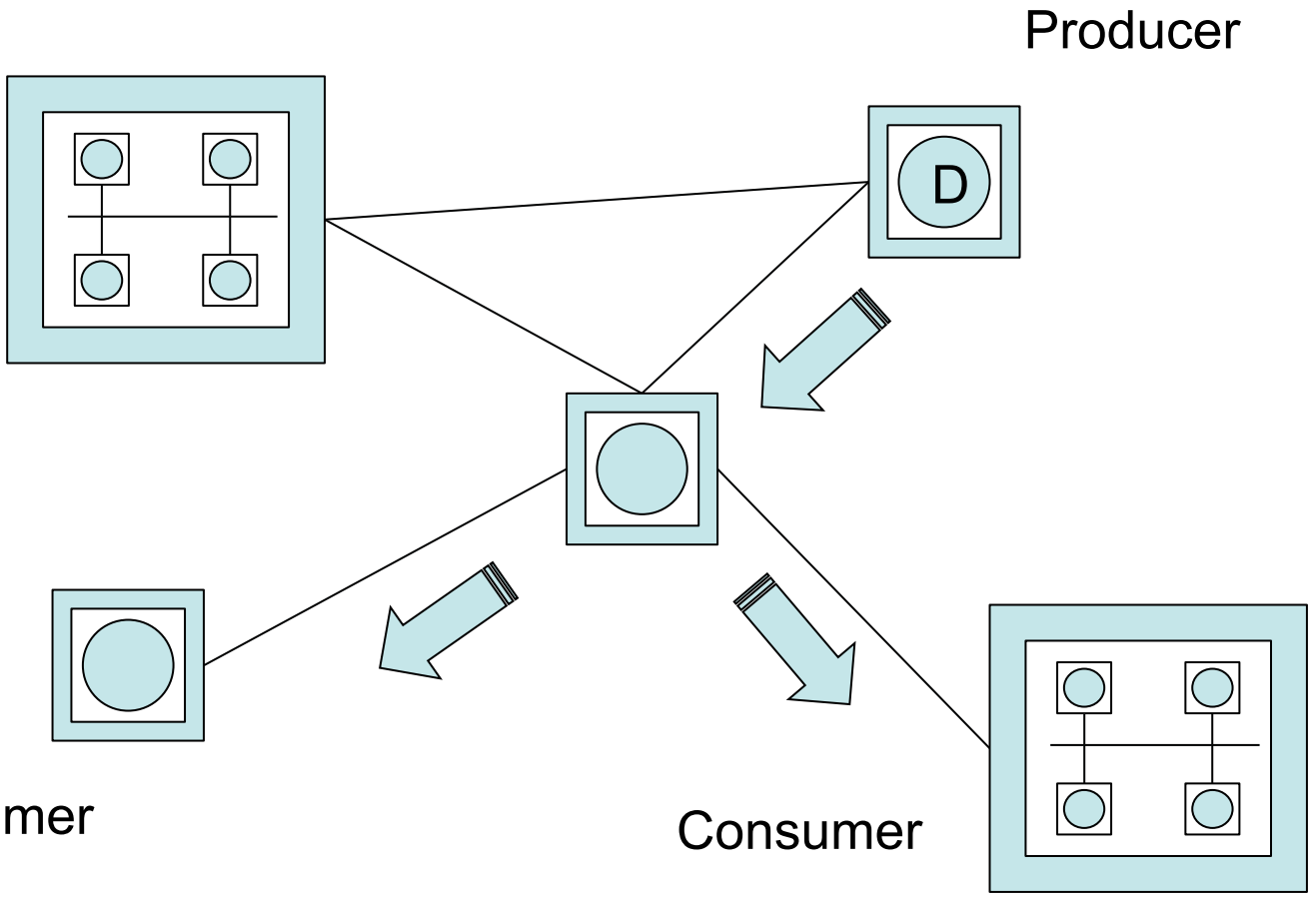
Testing, debugging, maintaining = Distributed OS/Middleware

DC Principles : Outline

- Introduction
  - What is Distributed Computing ?
  - Why distributing ?
  - What are the problems in distributing ?
- From theory …
- … to practice

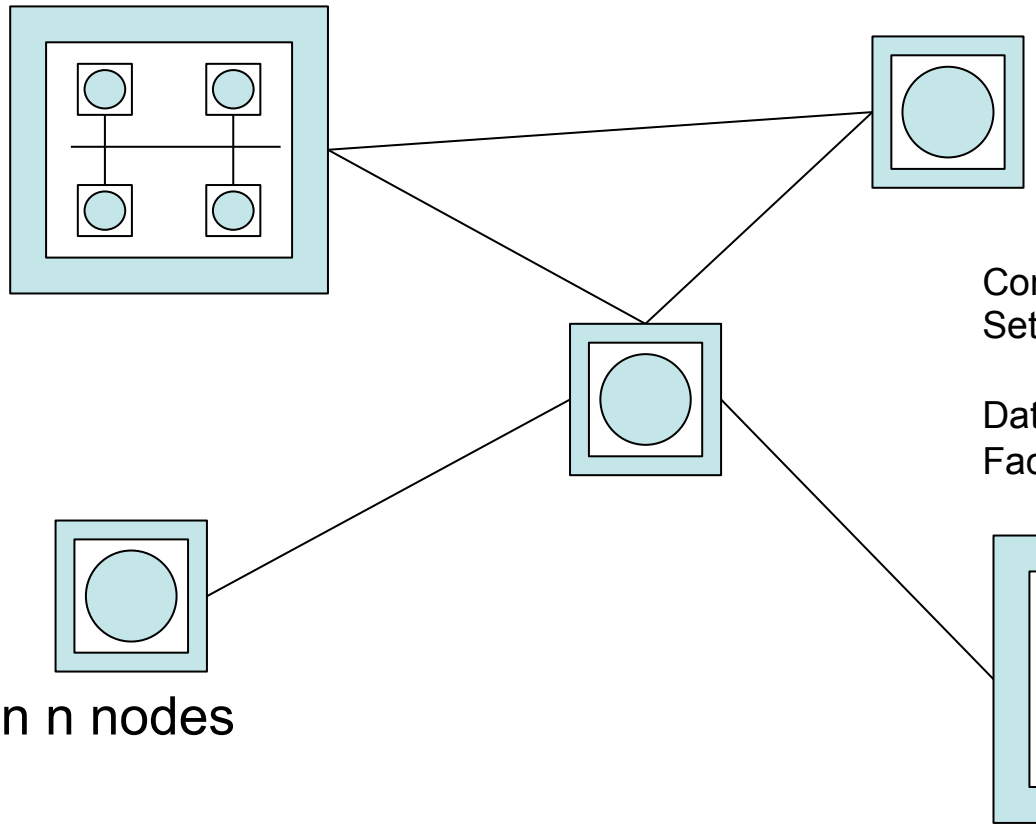
Why distributing ?

Application truly geographically distributed



Why distributing ?

Performance gain in
- computation
- storage



Computing Intensive:
Seti|Folding@Home

Data Intensive:
Facebook, Google

Speedup
$S_n = T_1 / T_n$

Efficiency
$E_n = T_1 / (n * T_n)$

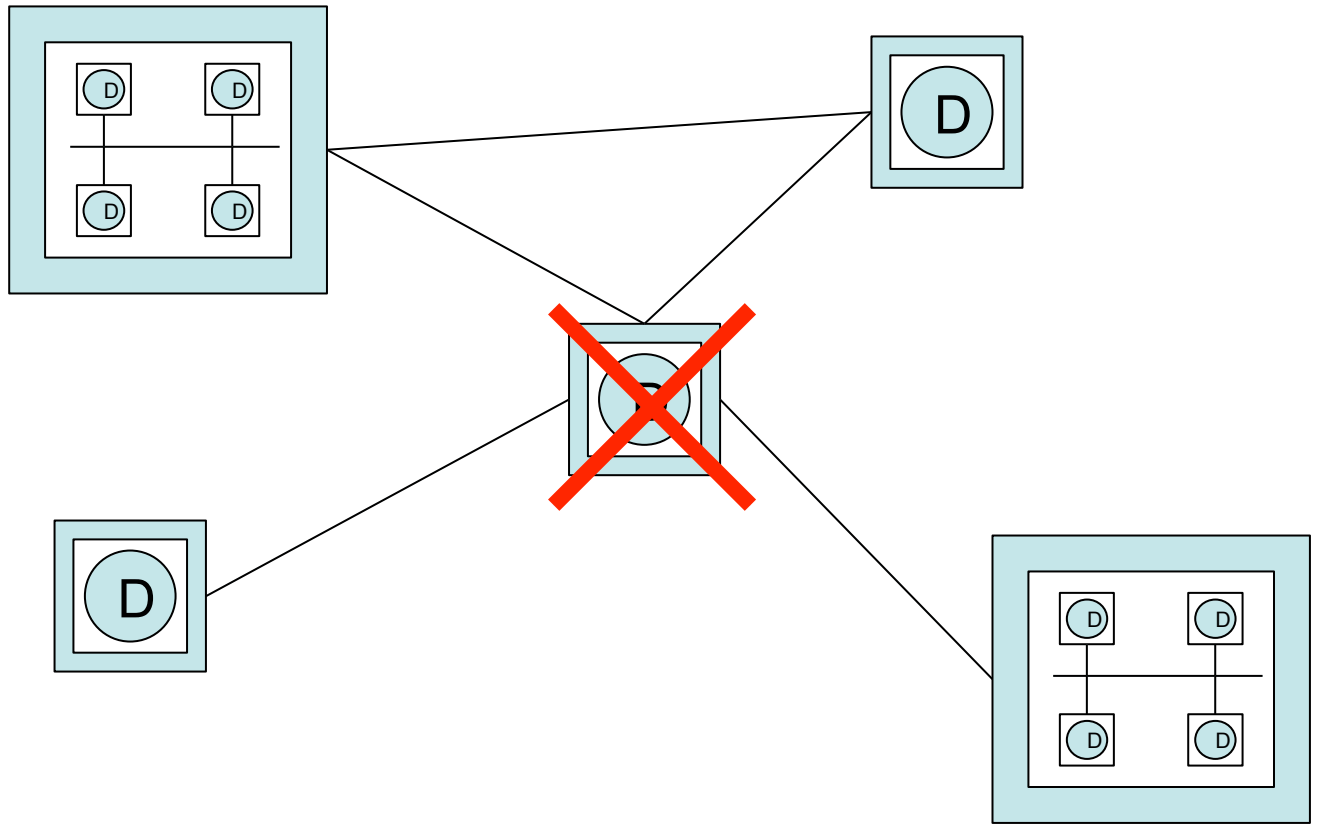
T_n: execution time on n nodes
[Eager 1989]

Why distributing ?

Robustness:
to guaranty
fault-tolerance in

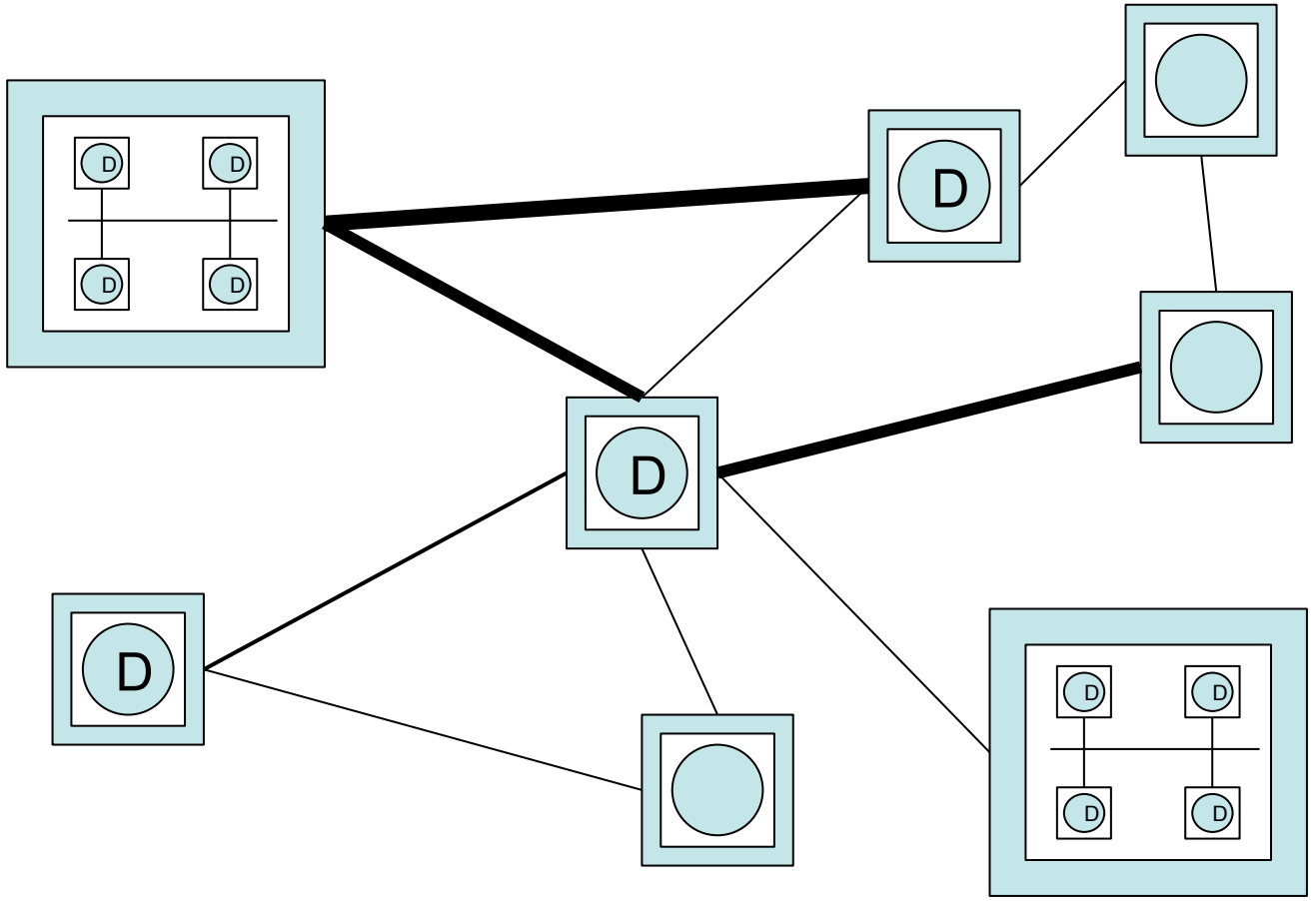
- computation
- storage

No Single Point
Of Failure (SPOF) :
other nodes can
execute/host the
same task/data
executed/hosted on
the failed node



What are problems in distributing ?

Dynamic Topology: the structure of the system is not know in advance and can change during execution



Number of nodes

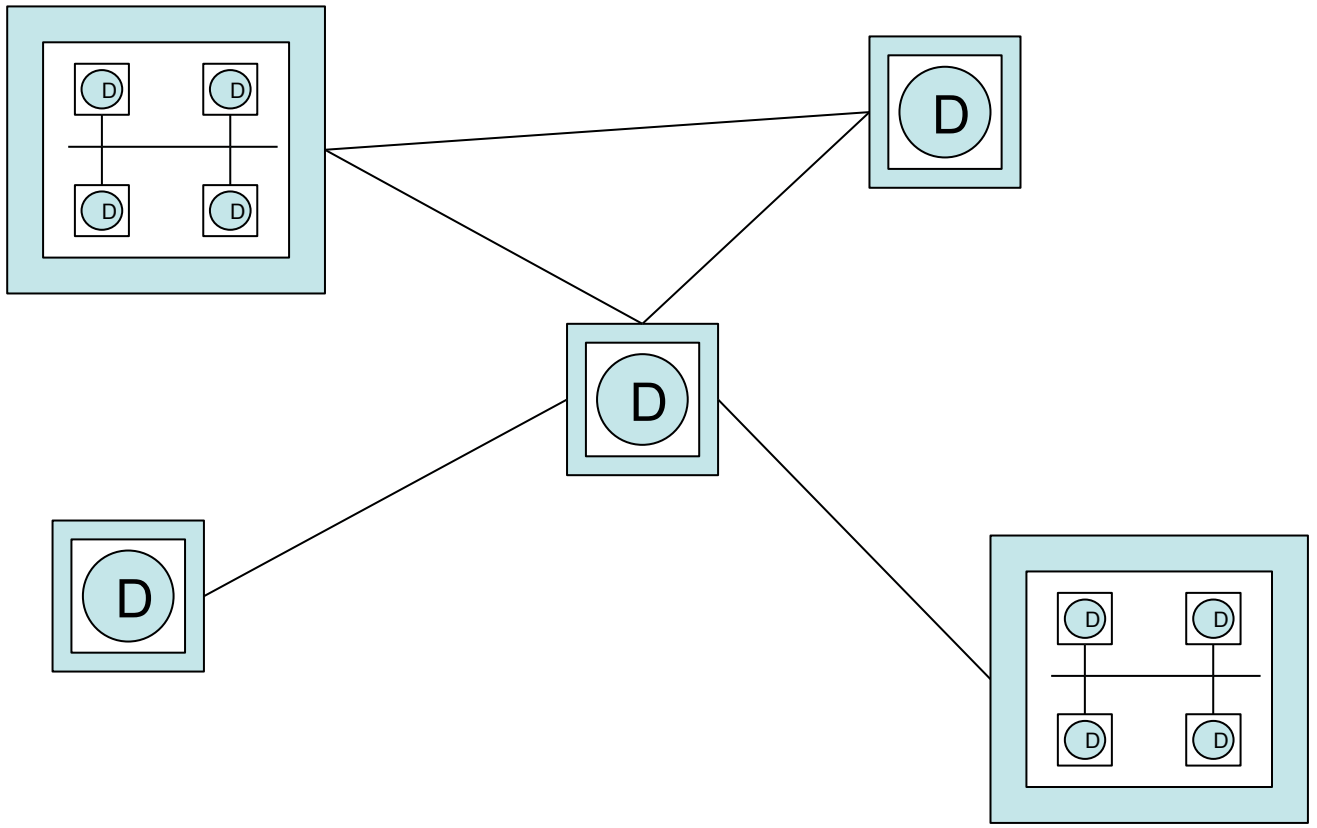
Number of links

Links characteristics

What are problems in distributing ?

Byzantine Faults:

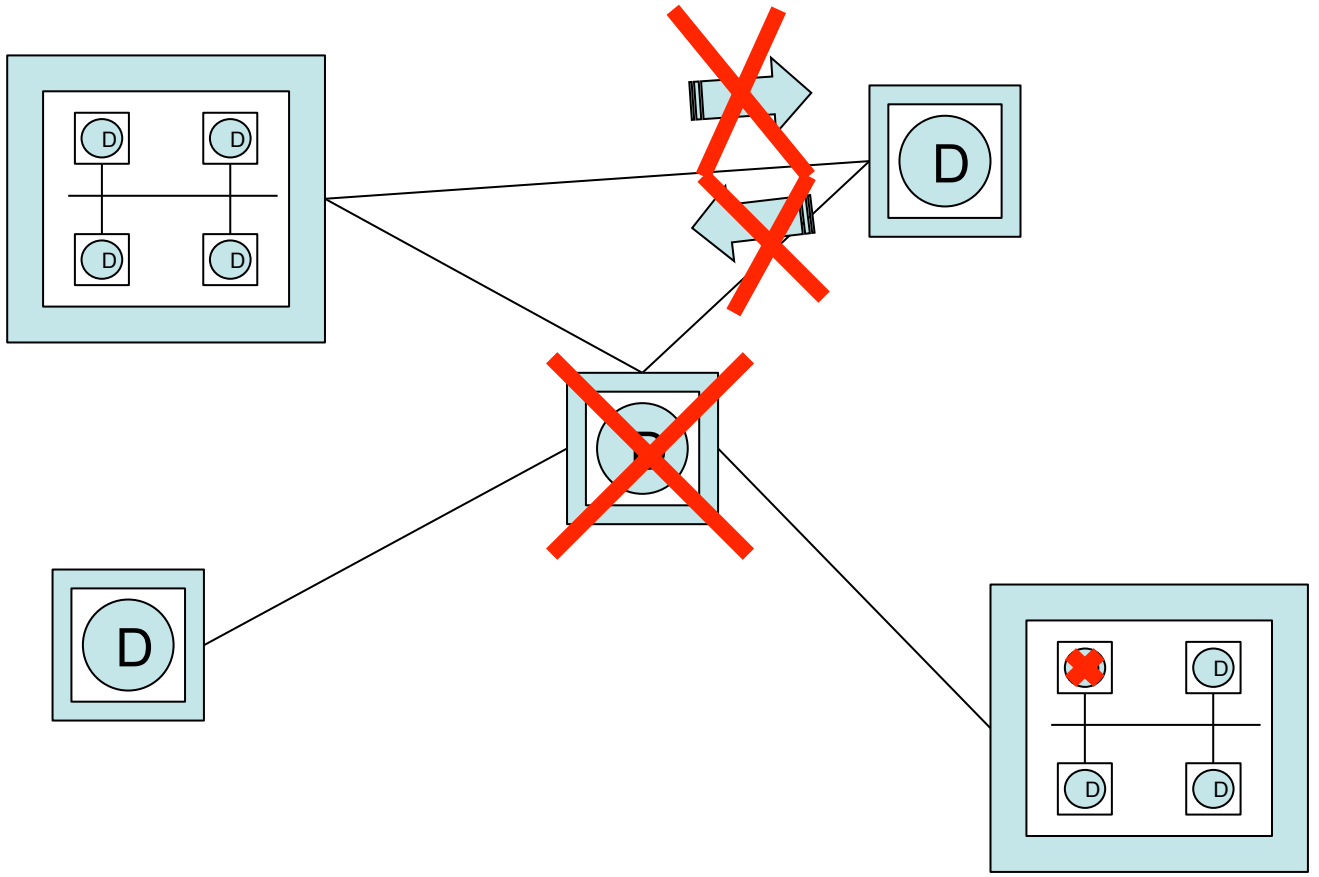
- Omission Failures
- Commission Failures



What are problems in distributing ?

Byzantine Faults:

- **Omission Failures**
- Commission Failures



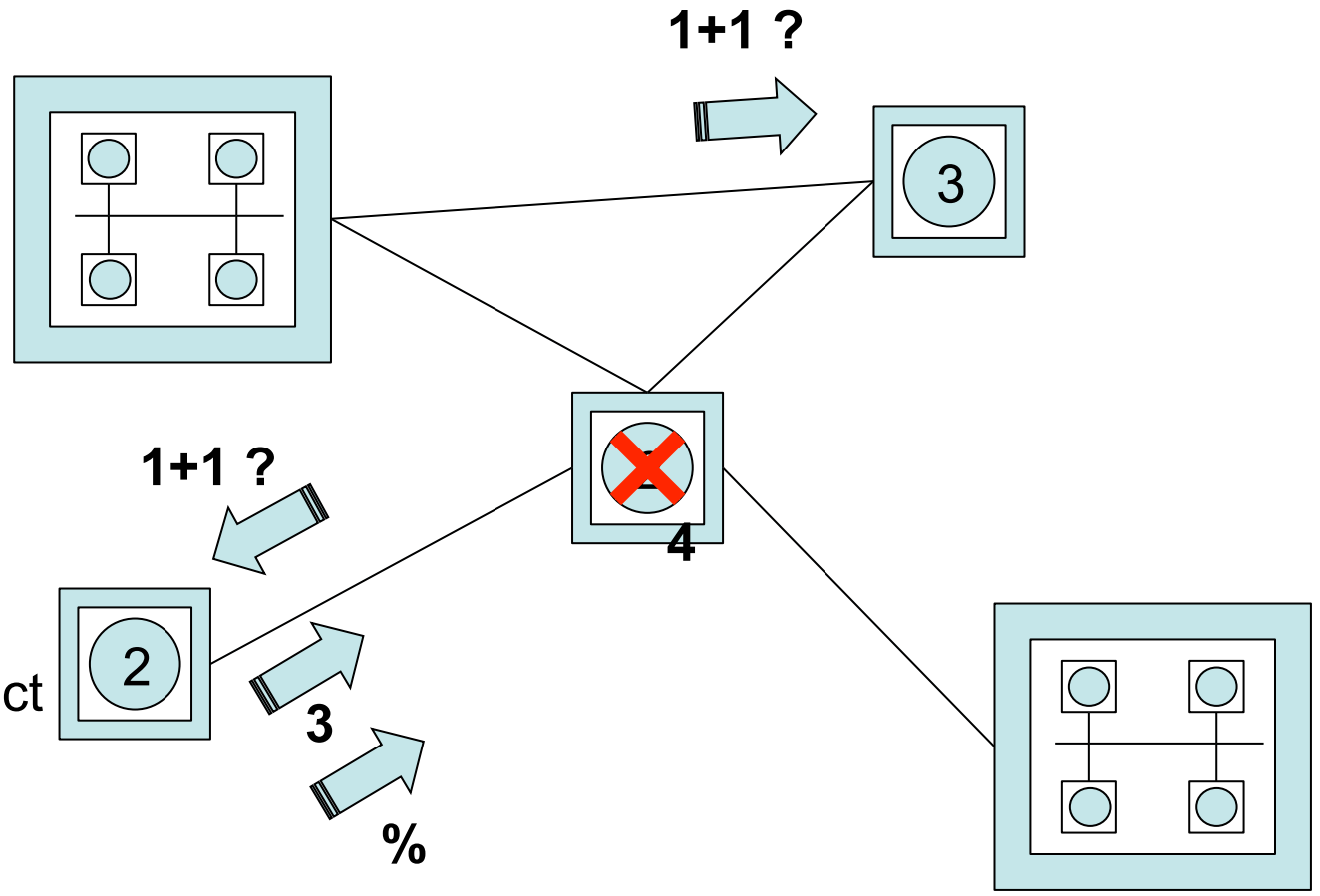
- Crash Failures
- Failing to receive a request
- Failing to send a response

What are problems in distributing ?

Byzantine Faults:

- Omission Failures
- **Commission Failures**

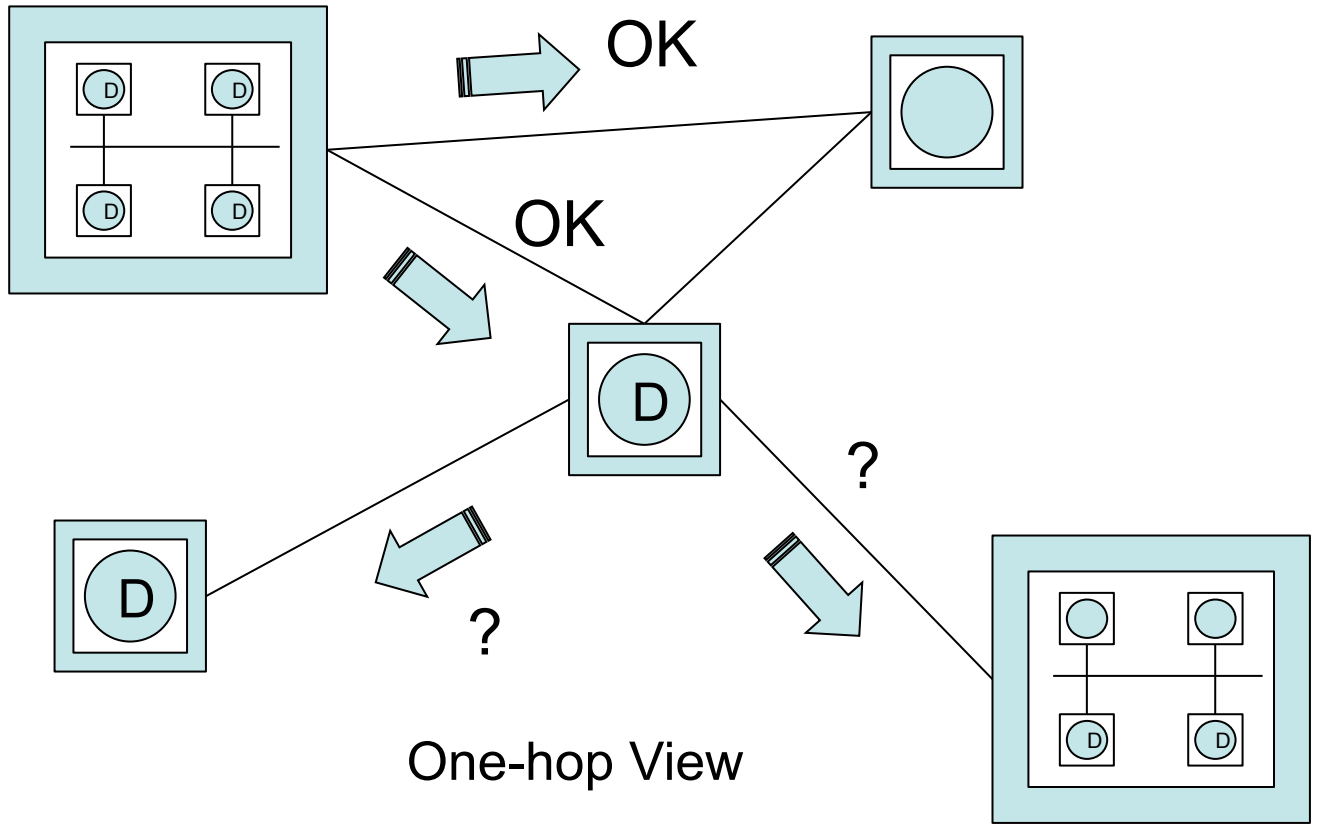
- Processing a request incorrectly
- Corrupting local state
- Sending an incorrect or inconsistent response to a request



What are problems in distributing ?

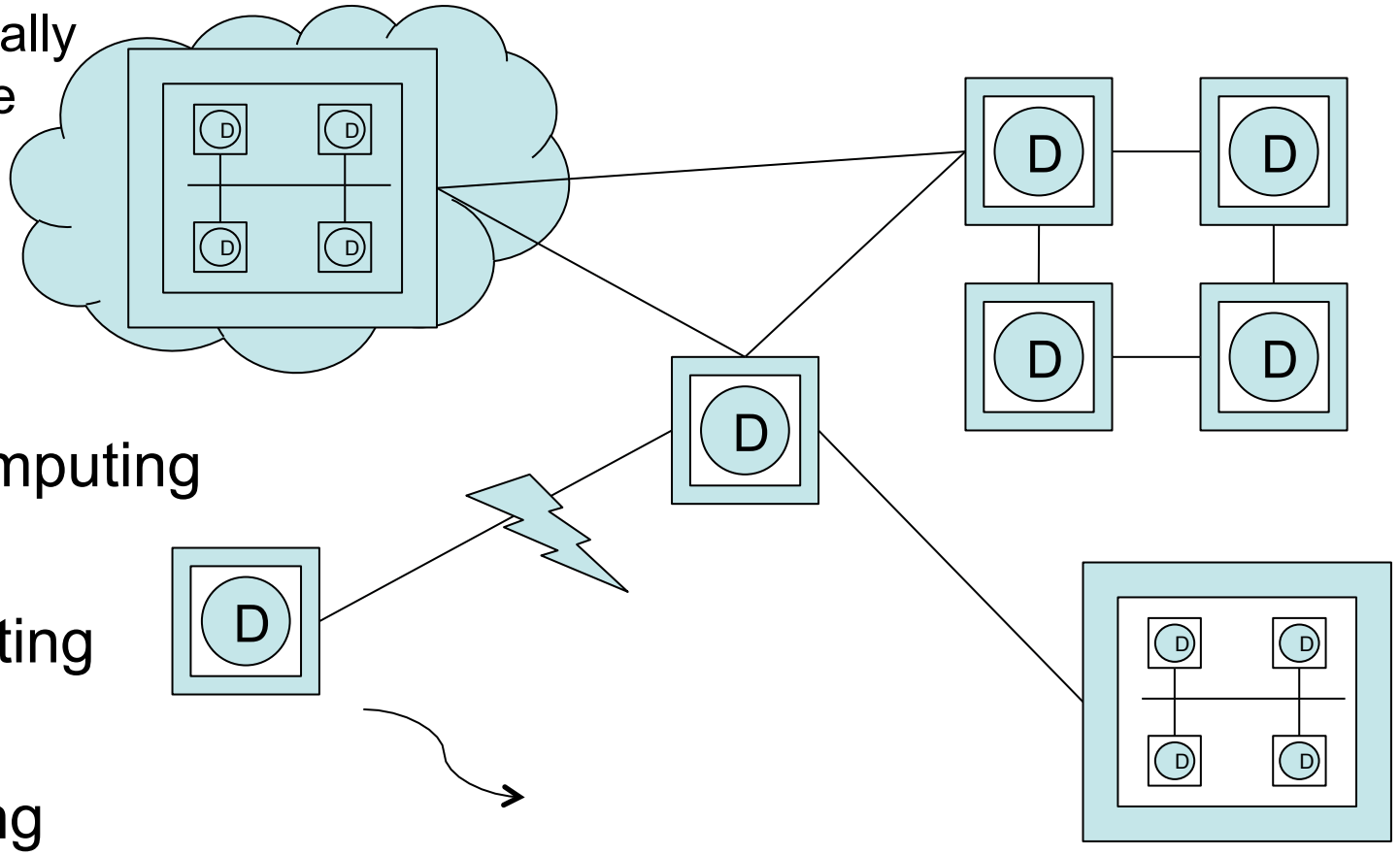
Decision-making: each autonomous node has an incomplete view of the system that leads to uncertain decisions

For scalability reasons, trade-off required between optimal decision and performance [Hayashibara 2002]



What are problems in distributing ?

Problems typically depends on the application domain



Section 4:
Pervasive Computing

Section 5:
Cloud Computing

Section 6:
Grid Computing

DC Principles : Outline

- Introduction
  - What is Distributed Computing ?
  - Why distributing ?
  - What are the problems in distributing ?
- From theory …
  - Distributed Algorithms:
    - Graph colouring, Mutual exclusion,
    - Consensus, Byzantine fault-tolerance, Self-stabilization
  - Complexity

DC : Theory

- Theoretical Computer Science
  - Computational Problem:
    - An entity (human-being or other) asks a question (Input)
    - A computer performs some computation
    - The computer produces an answer (Output)

DC : Theory

- Theoretical Computer Science
  - Computability Theory:

⇔Which problems are computational ?

⇔Which problems can be solved using a computer ?

⇔Which problems can be designed by an **algorithm** and executed by Random Access Machine or a Universal Turing Machine ?

DC : Theory

- Theoretical Computer Science
  - Computational Complexity Theory: How efficient is the algorithm that solves the problem ?

DC : Theory

- and Distributed Computing ?
  - Computational Problem:
    - An entity (human-being or other) asks a question (Input)
    - **Several computers** perform **some computations and interactions**
    - These computers produce an answer (Output)

DC : Theory

- and Distributed Computing ?
  - Computability Theory:

  ⇔Which problems are computational ?

  ⇔Which problems can be solved using a **network of computers** ?

  ⇔What is the concurrent or distributed equivalent of a sequential general-purpose computer ?

DC : Theory

- and Distributed Computing ?
  - Computational Complexity Theory: How efficient is the concurrent or distributed algorithm that solves the problem ?

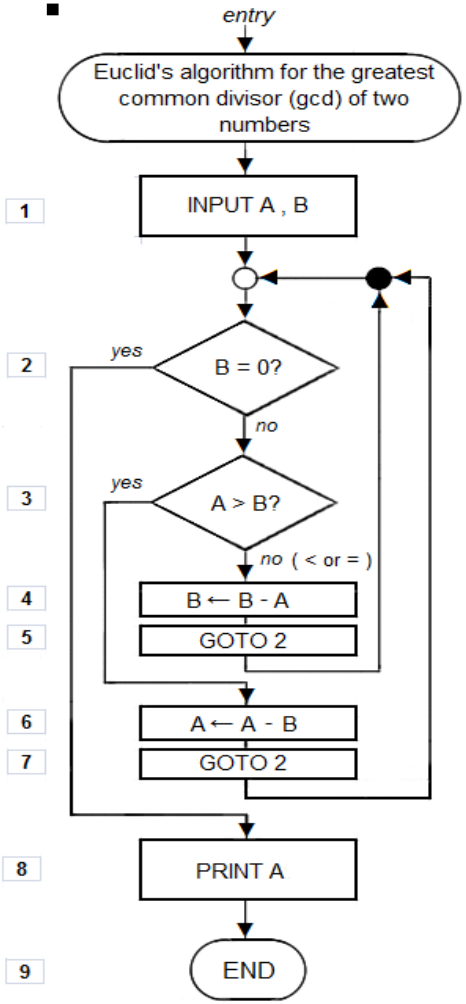
Speedup
$S_n = T_1 / T_n$

Efficiency
$E_n = T_1 / (n * T_n)$

T_n: execution time on n nodes
[Eager 1989]

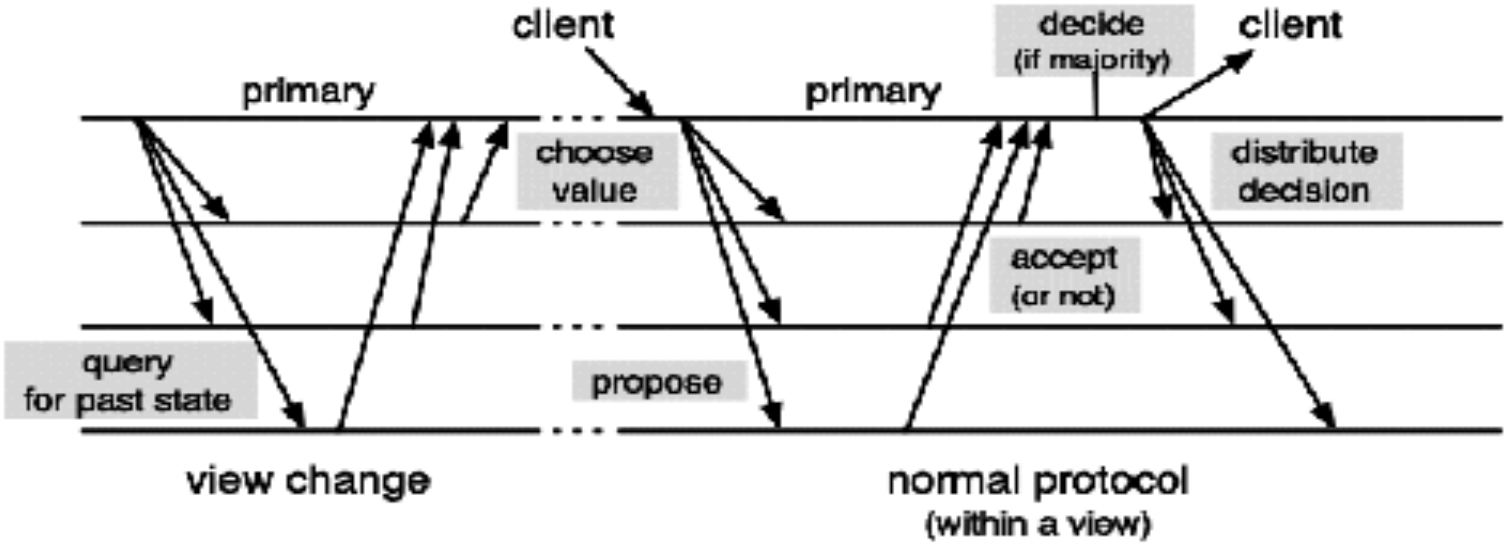
DC : Algorithm ?

« An algorithm is an effective method expressed as a **finite** list of **well-defined instructions** for **calculating a function**. […] In simple words an algorithm is a step-by-step procedure for calculations. » [wikipedia/AL]



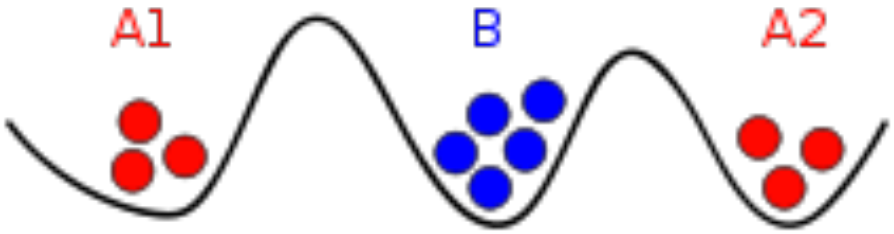
DC : Distributed Algorithm ?

« A distributed algorithm is an algorithm designed to run on computer hardware constructed from **interconnected processors** » [wikipedia/DA]



Paxos Consensus Protocol [Lamport 1998]

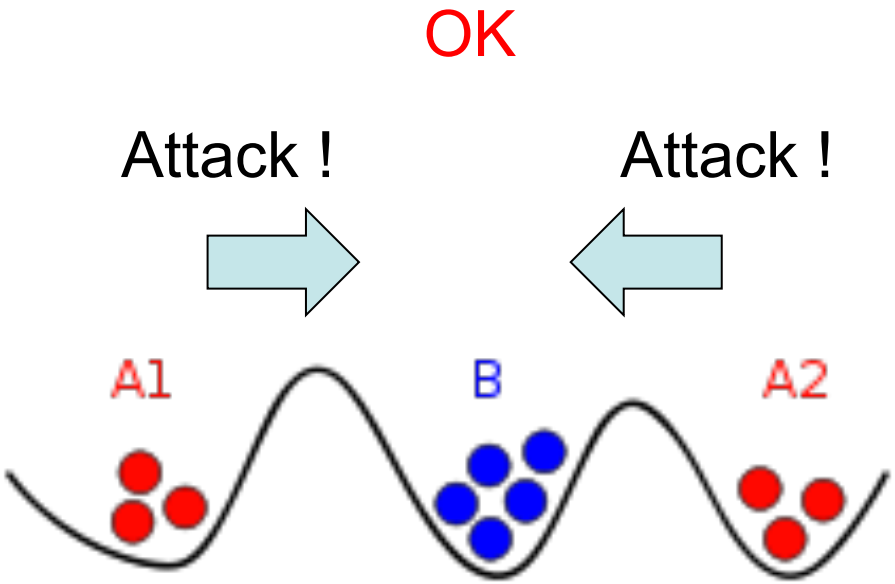
DC : Byzantine Generals Problem



All problems comes from … interconnected processors, truly ?

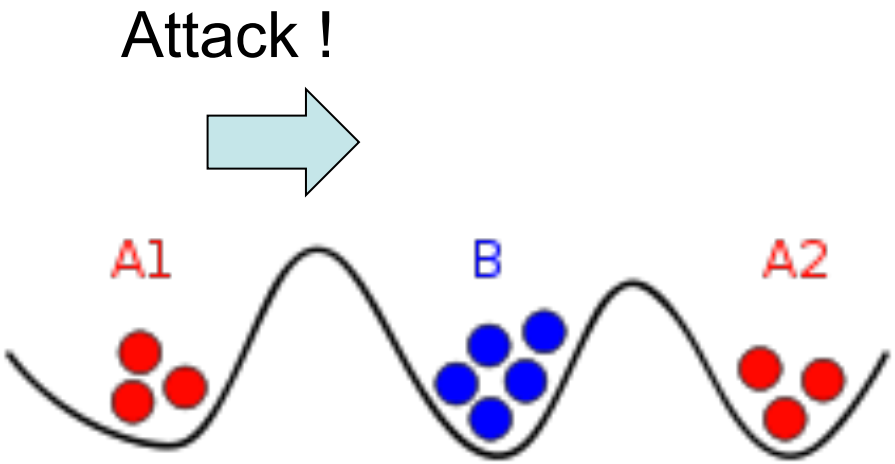
« Two armies, each led by a general, are preparing to attack a fortified city. The armies are encamped near the city, each on its own hill. A valley separates the two hills, and the only way for the two generals to communicate is by sending messengers through the valley. Unfortunately, the valley is occupied by the city's defenders and there's a chance that any given messenger sent through the valley will be captured. Note that while the two generals have agreed that they will attack, they haven't agreed upon a time for attack before taking up their positions on their respective hills. »

DC : Byzantine Generals Problem



DC : Byzantine Generals Problem

FAIL!

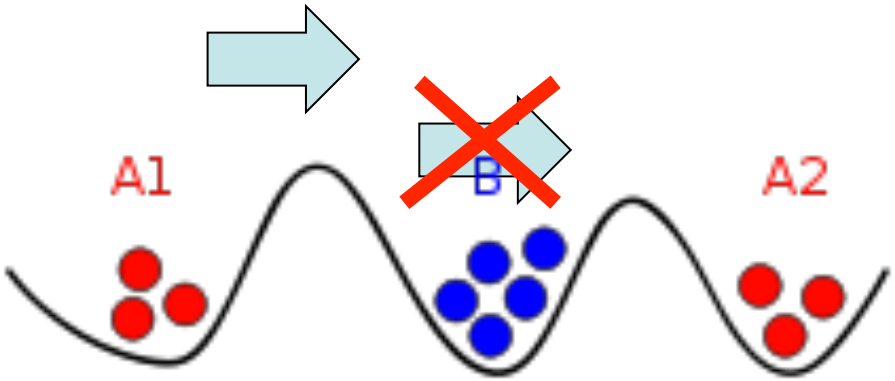


DC : Byzantine Generals Problem

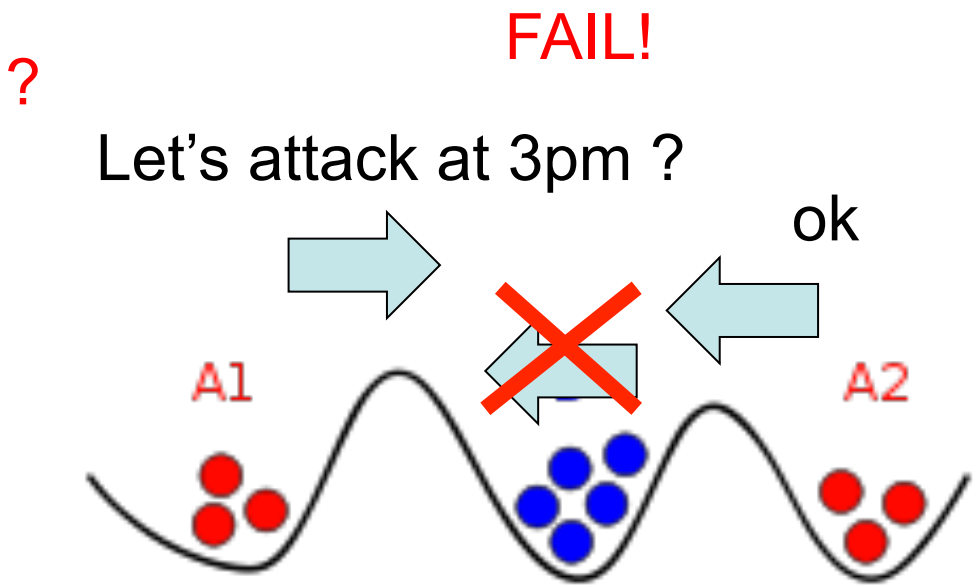
FAIL!

?

Let’s attack at 3pm ?



DC : Byzantine Generals Problem



No solution to the problem [Akkoyunlu 1975]
and can even be more complex: n generals, traitors, etc.

Solution to generate BFT protocols [Guerraoui 2010]

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Synchronizers
  - Logical Clock
  - Clock Synchronization

Time is also a data !

  - Atomic commit
  - Replication

DC : Classic Distributed Algorithms

- Resource allocation
  - Mutual exclusion
  - Graph colouring

- Decision-making
  - Leader election
  - Consensus

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Synchronizers
  - « A Synchronizer is an algorithm that can be used to run a synchronous algorithm on top of asynchronous processor network » [Awerbuch 1985]
    - Alpha/Beta/Gamma synchronizer

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Synchronizer : Alpha Synchronizer
  - In round r, the synchronizer at p sends p's message (tagged with the round number) to each neighbor p' or *no-msg(r)* if it has no messages. When it collects a message or no-msg from each neighbor for round r, it delivers all the messages.
  - Allow to model simplified “ideal network”

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Logical Clock is a mechanism for capturing chronological and causal relationships in a distributed system
    - Lamport timestamps
    - Vector clocks
    - Version vectors
    - Matrix clocks

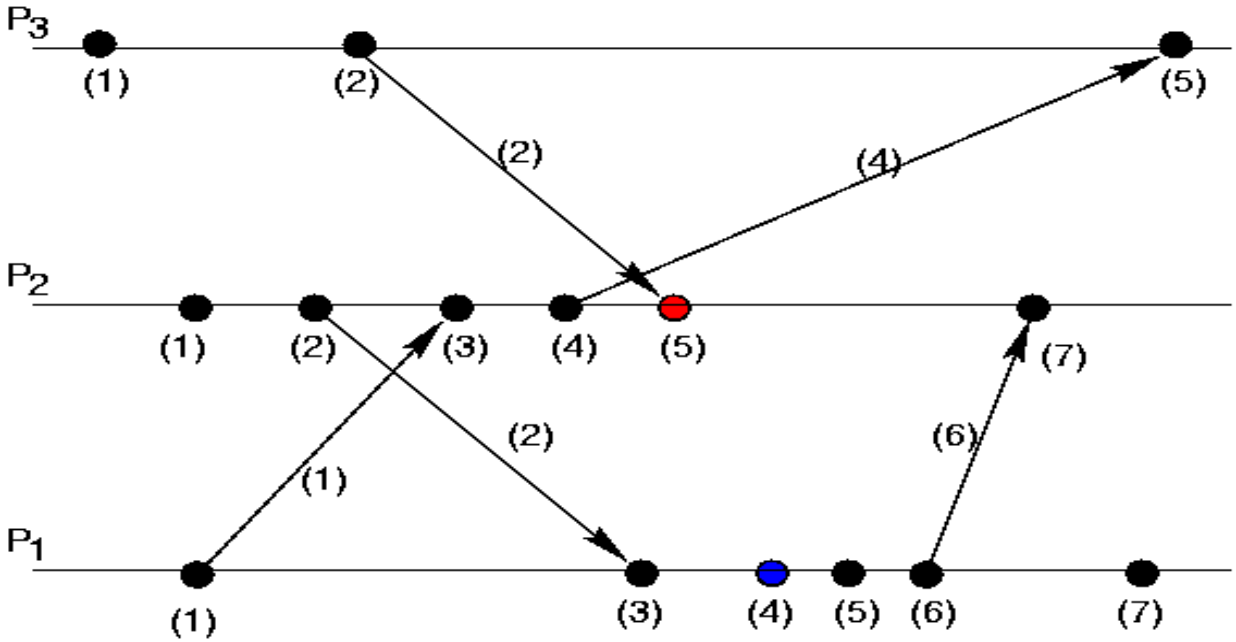
DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Logical Clock: Lamport timestamps [Lamport 1978]

  - A process increments its counter before each event in that process;
  - When a process sends a message, it includes its counter value with the message;
  - On receiving a message, the receiver process sets its counter to be greater than the maximum of its own value and the received value before it considers the message received.

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Logical Clock: Lamport timestamps [Lamport 1978]



DC : Classic Distributed Algorithms

- Data validation ➔ Probability
  - Clock Synchronization: Internal clocks of different computers differs because of the clock drift

- Trivial algorithms in a centralized approach [Cristian 1989] [Gusella 1989] :
  - P requests the time from S
  - After receiving the request from P, S prepares a response and appends the time T from its own clock
  - P then sets its time to be T + RTT/2

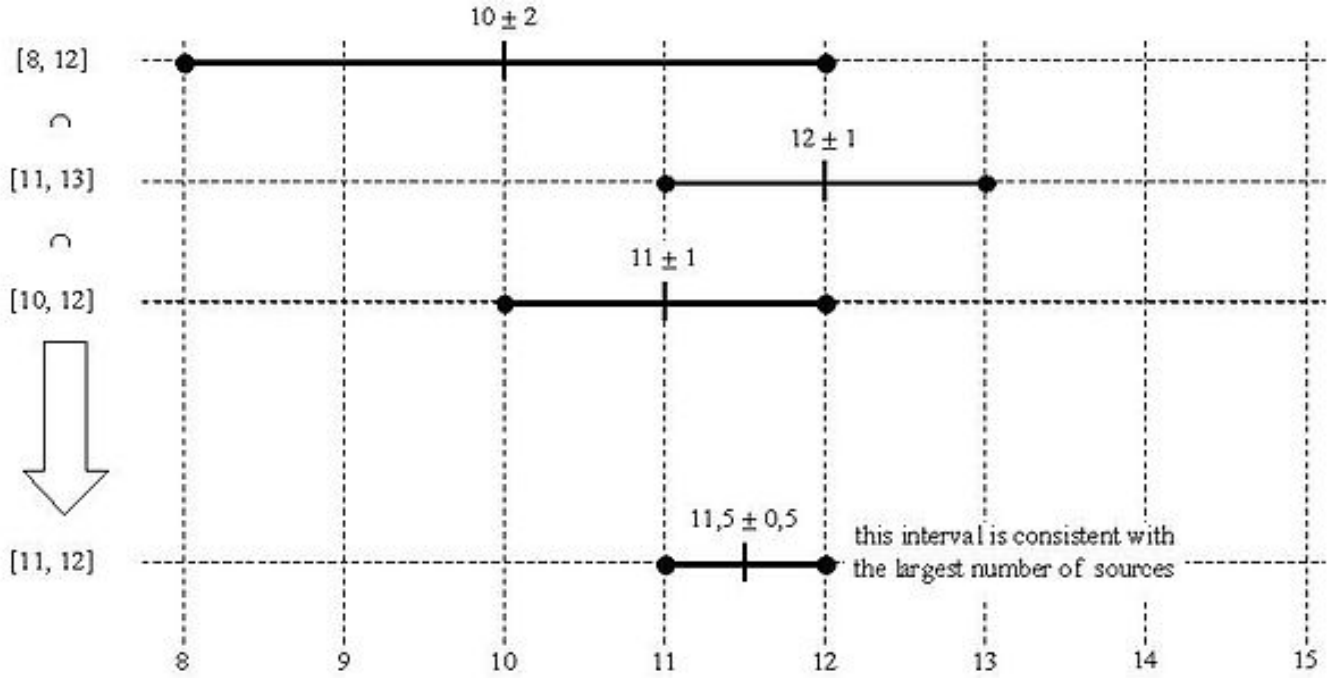
DC : Classic Distributed Algorithms

- Data validation ➔ Probability
  - Clock Synchronization: Internal clocks of different computers differs because of the clock drift

- In a distributed network, Marzullo's algorithm [Marzullo 1984] is an agreement algorithm used to select sources for estimating accurate time from a number of noisy time sources
  - A refined version, “the intersection algorithm” is used in NTP [Mills 2006]

DC : Classic Distributed Algorithms

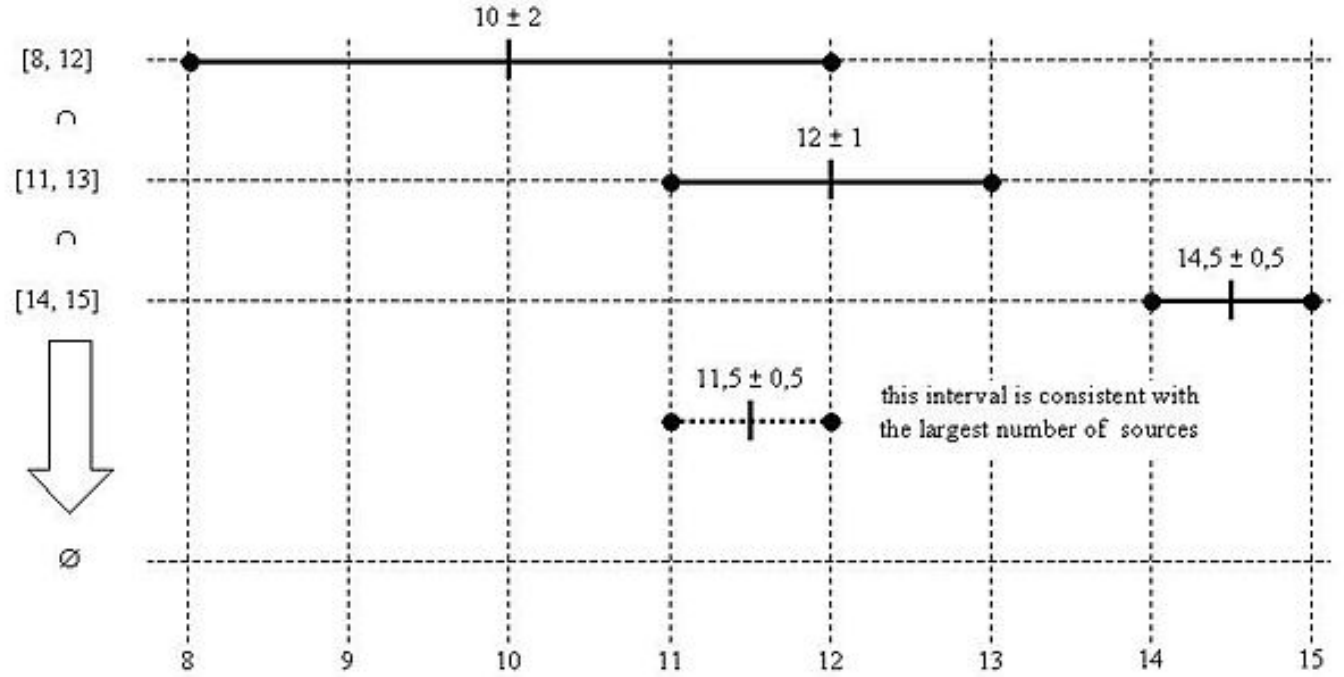
- Data validation ➔ Probability
  - Clock Synchronization: intersection algorithm



[wikipedia/MA]

DC : Classic Distributed Algorithms

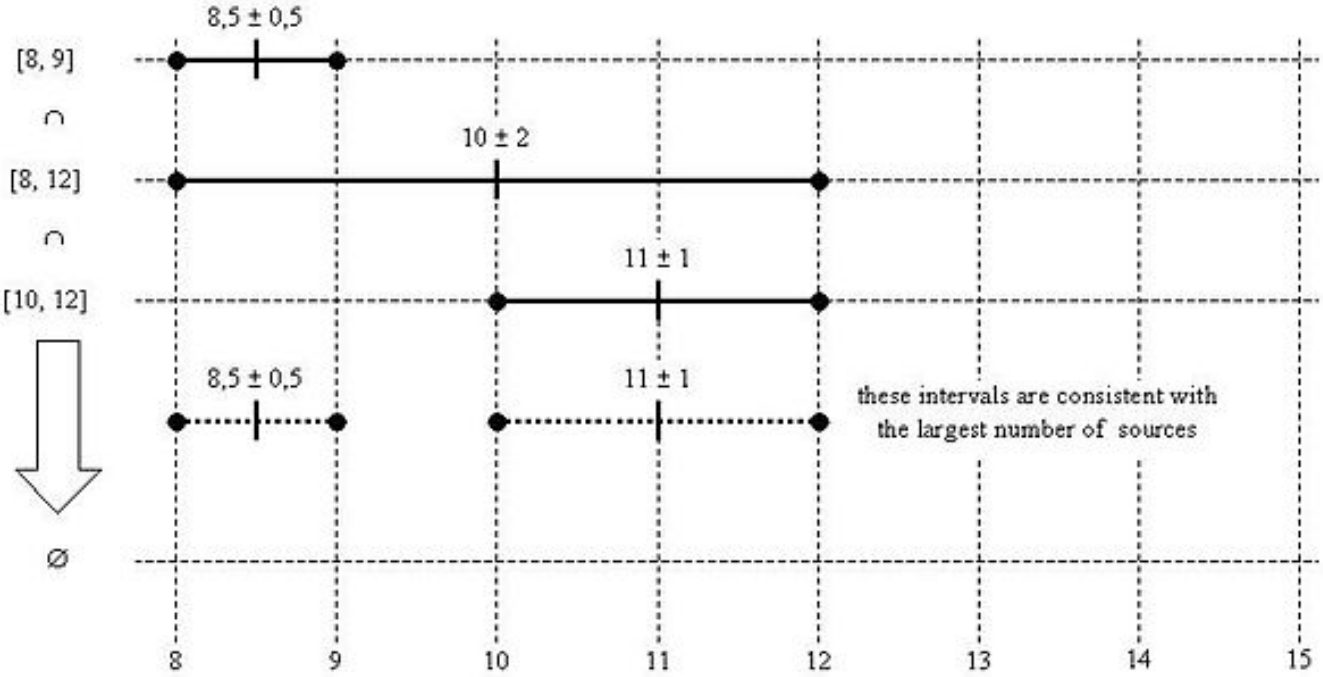
- Data validation ➔ Probability
  - Clock Synchronization: intersection algorithm



[wikipedia/MA]

DC : Classic Distributed Algorithms

- Data validation ➔ Probability
  - Clock Synchronization: intersection algorithm



[wikipedia/MA]

DC : Classic Distributed Algorithms

- Data validation ➔ Probability
  - Clock Synchronization: depending on network topology
    - CS-MNS: Clock Sampling Mutual Network Synchronization is suitable for distributed and mobile applications [Rentel 2005]
    - Reference broadcast synchronization is often used in wireless networks and sensor networks. An initiator broadcasts a reference message to urge the receivers to adjust their clocks [Ganeriwal 2003] [Maroti 2004]
    - PTP: Precision Time Protocol is used in measurement and control systems in local area networks and can achieve sub-microsecond accuracy [IEEE 1588]

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Atomic commit

  - An atomic commit is an operation where a set of distinct changes is applied as a single operation. If the atomic commit succeeds, it means that all the changes have been applied. If there is a failure before the atomic commit can be completed, the "commit" is aborted and no changes will be applied.

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Atomic commit : two-phase algorithm

- The coordinator sends a query to commit message to all cohorts and waits until it has received a reply from all cohorts.
- The cohorts execute the transaction up to the point where they will be asked to commit. They each write an entry to their *undo log* and an entry to their *redo log*
- Each cohort replies with an agreement message (cohort votes Yes to commit), if the cohort's actions succeeded, or an abort message (cohort votes No, not to commit), if the cohort experiences a failure that will make it impossible to commit.

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Two-phase algorithm
  - Pb: blocking algo

A single node will continue to Wait even if all other sites have failed

If the coordinator fails permanently, some cohorts will never resolve their transactions

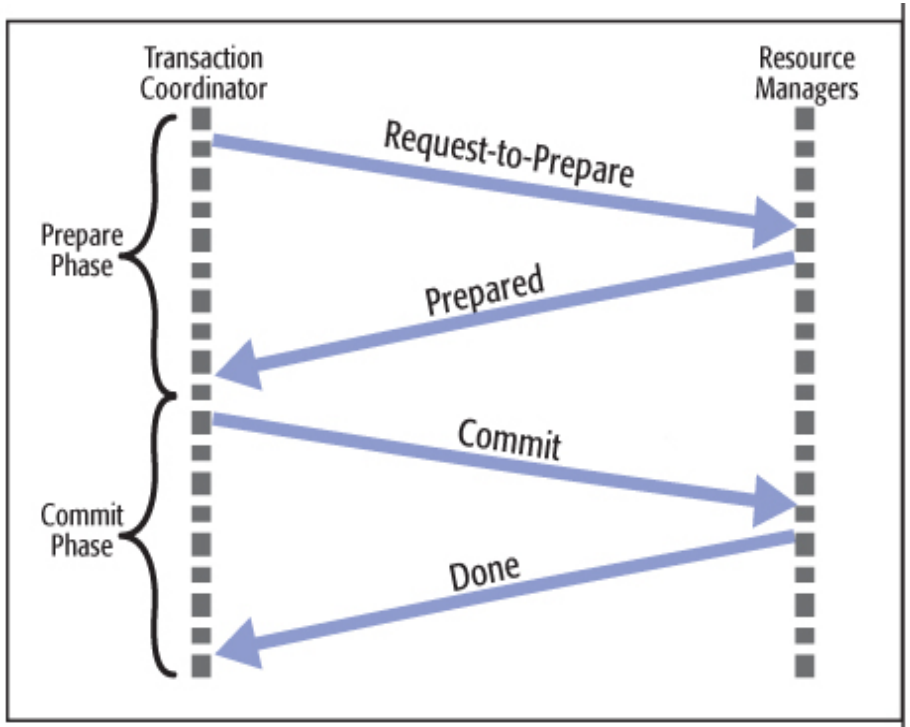
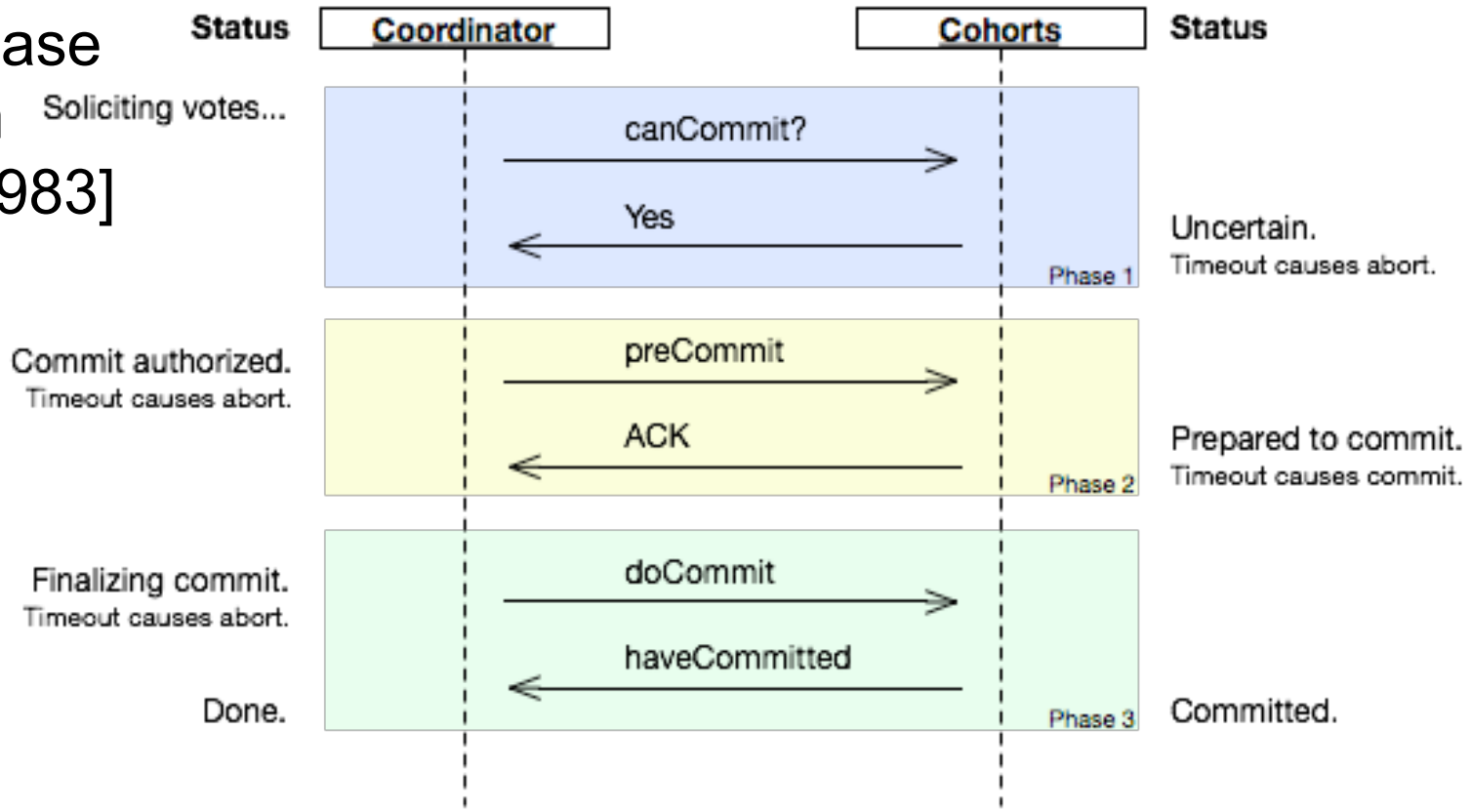


Figure 1 • The two-phase commit protocol

DC : Classic Distributed Algorithms

- Three-phase algorithm [Skeen 1983]



[wikipedia]

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication :

« Replication is the process of sharing information so as to ensure **consistency** between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility. It could be *data replication* if the same data is stored on multiple storage devices, or *computation replication* if the same computing task is executed many times. A computational task is typically *replicated in space*, i.e. executed on separate devices, or it could be *replicated in time*, if it is executed repeatedly on a single device. » [wikipedia/R]

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : two kinds
  - Active replication : each request is executed on each replica and each of these modifications are transferred to all replica
  - Passive replication : each request is executed on a single replica and this modification is transferred to all replica (*master-slave*)

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : three algorithms models
  - Transactional replication (active/passive) : transactions on replicated data in accordance with ACID properties (*atomicity, consistency, isolation, durability*)
  - (slow, on copy-guaranty)

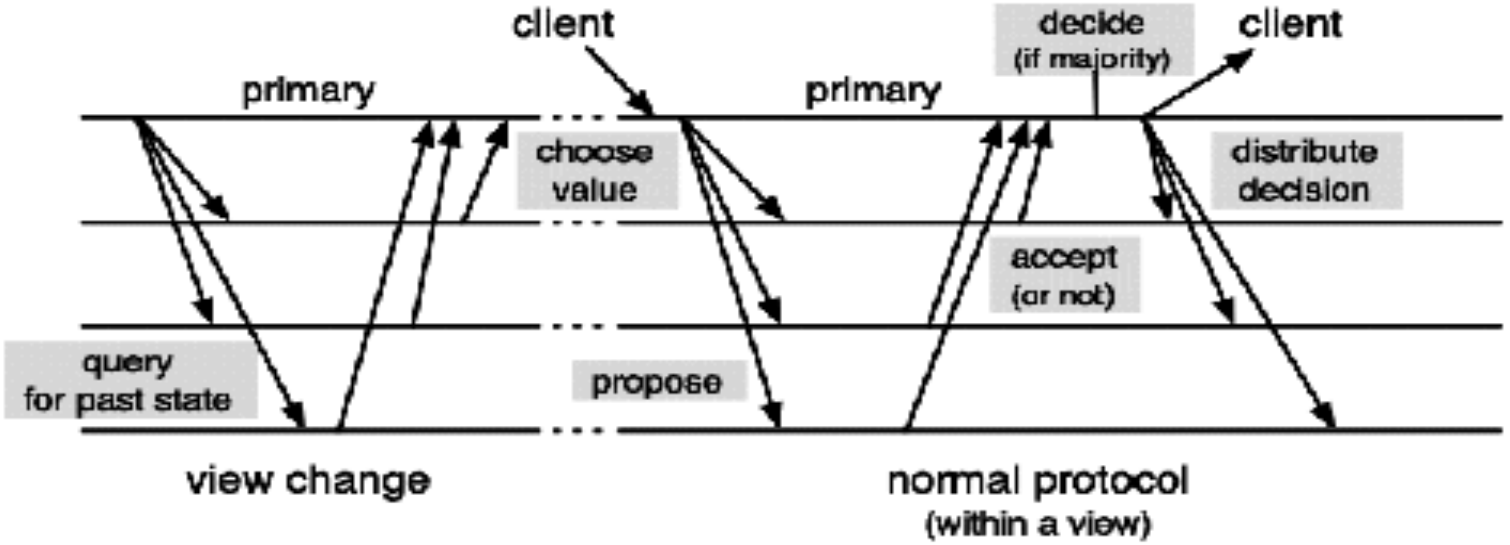
DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : three algorithms models
  - State machine replication (active) : deterministic finite state machine, atomic broadcast, distributed consensus
  - (mid-fast)

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : three algorithms models
  - State machine replication (active)

Paxos Consensus Protocol [Lamport 1998]



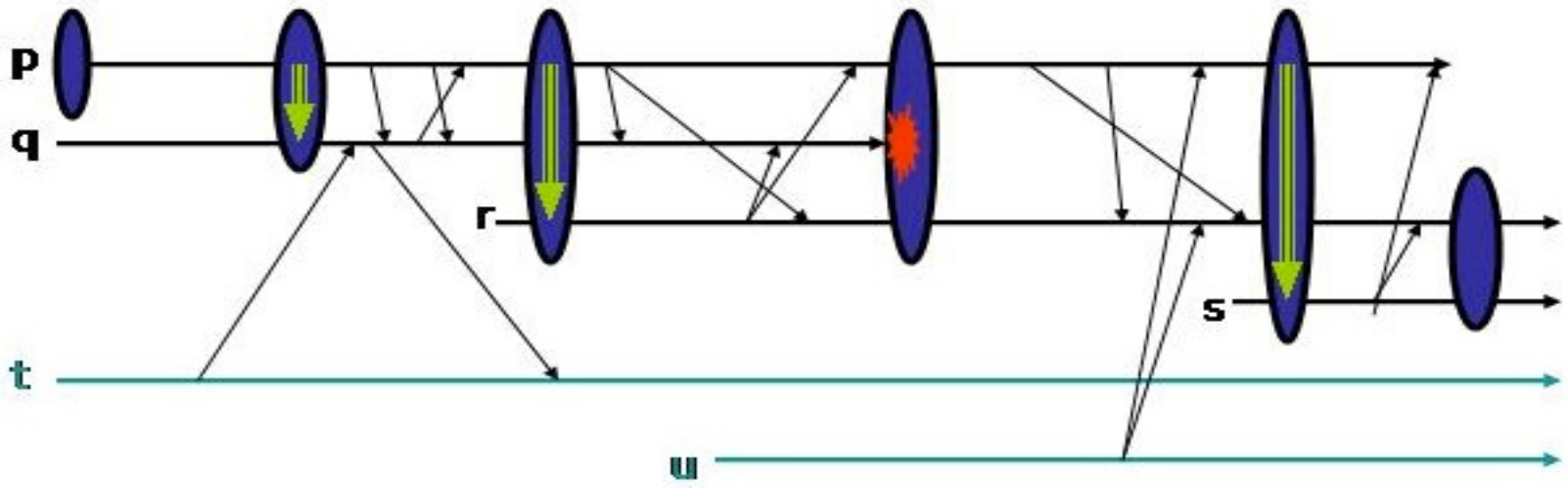
DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : three algorithms models
  - Virtual synchrony (active/passive) : process group, checkpoint, multicast
  - (fastest, less rigourous)

DC : Classic Distributed Algorithms

- Data validation ➔ Order
  - Replication : three algorithms models
  - Virtual synchrony (active/passive)

[Défago 2004]



DC : Classic Distributed Algorithms

- Resource allocation
  - Mutual exclusion
  - Graph colouring

- Decision-making
  - Leader election
  - Consensus

DC : Classic Distributed Algorithms

- Resource allocation
  - Mutual exclusion (Mutex) : algorithm to avoid simultaneous use of a common resource. This piece of code in which a process access a common resource is called a *critical section*.
  - Multiple use : locks, reentrant mutexes, semaphores, monitors, message passing, tuple space

DC : Classic Distributed Algorithms

- Resource allocation
  - Mutual exclusion (Mutex) : different algorithms [Raynal 1986]
  - Dekker's algorithm
  - Peterson's algorithm
  - Lamport's bakery algorithm
  - Szymanski's algorithm

DC : Classic Distributed Algorithms

- Resource allocation
  - Mutual exclusion (Mutex) : different algorithms
  - Dekker's algorithm in 1965 [E.W. Dijkstra 2009]

```
flag[0] := false
flag[1] := false
turn := 0   // or 1

flag[0] := true
 while flag[1] = true {
     if turn ≠ 0 {
          flag[0] := false
          while turn ≠ 0 {}
          flag[0] := true
     }
 }
// critical section
...
turn := 1
flag[0] := false
```

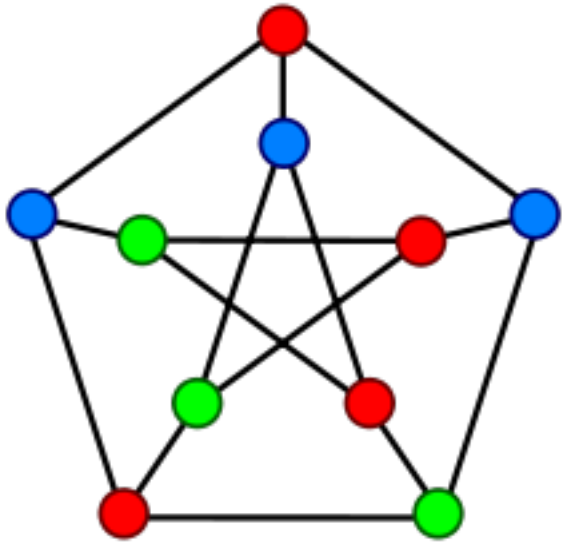
DC : Classic Distributed Algorithms

- Resource allocation
  - Graph colouring :

« Graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring. » [wikipedia/GC]

DC : Classic Distributed Algorithms

- Resource allocation
  - Graph colouring :
  - Multiple use : scheduling, bandwidth allocation, compilation register allocation, pattern matching, Sudoku ☺ (9-coloring on given specific graph with 81 vertices)



DC : Classic Distributed Algorithms

- Resource allocation
  - Graph colouring : Multi-trials technique allows to break symmetry efficiently [Schneider 2010] : randomized fastest algorithm that employs more attempts with every message exchange
  - Idea : In classical algorithms, every uncolored node randomly picks an available color and keeps it if no neighbor (concurrently) chooses the same color. For the multi-trials technique, a node gradually increases the number of chosen colors in every communication rounds.

DC : Classic Distributed Algorithms

- Decision-making
  - Leader election

« Leader election is the process of designating a single process as the organizer of some task distributed among several computers (nodes). » [wikipedia/LE]

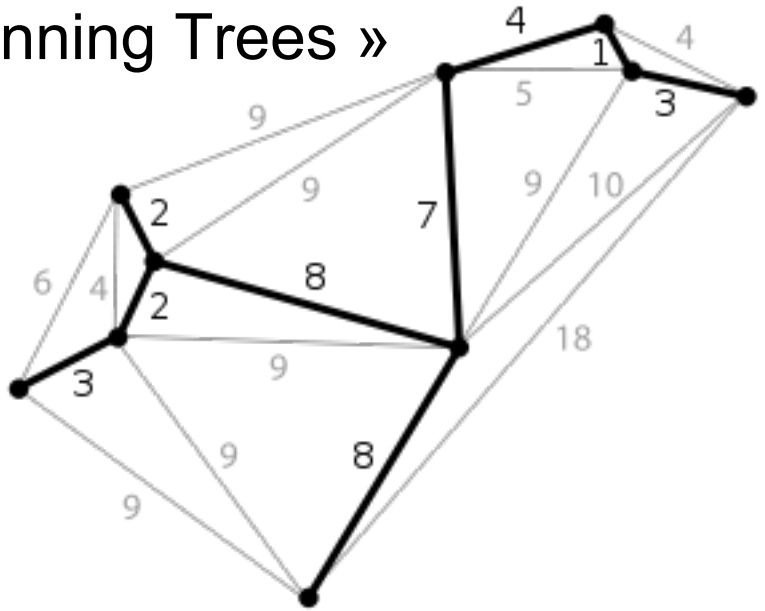
DC : Classic Distributed Algorithms

- Decision-making
  - Leader election : different algorithms [Korach 1990]
  - Bully algorithm : higher process ID
  - Chang and Roberts algorithm : UID, token ring
  - Gallager, Humblet, and Spira algorithm
  - Voting System

DC : Classic Distributed Algorithms

- Decision-making
  - Leader election : Gallager, Humblet, and Spira algorithm [Galager 1983] : « Distributed Algorithm for Minimum-Weight Spanning Trees »

A minimum spanning tree (MST) or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree



DC : Classic Distributed Algorithms

- Decision-making
  - Consensus

"Consensus is the problem of task of group agreement in the presence of faults."

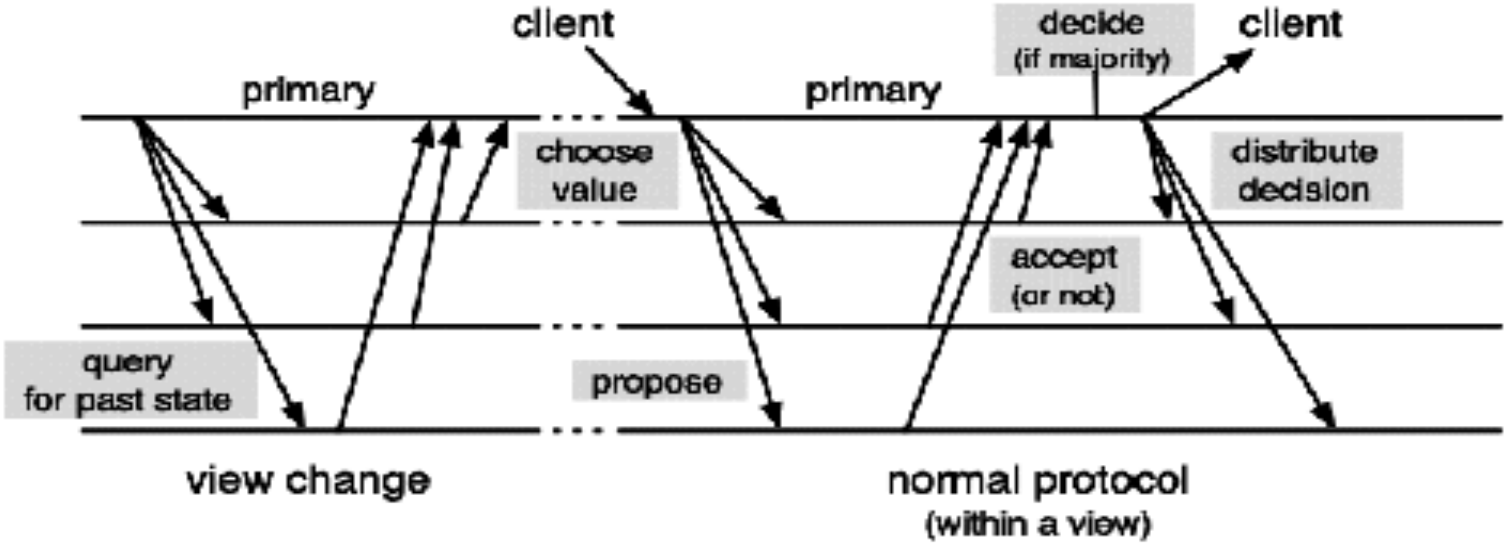
Consensus has been shown to be impossible to solve in several models of distributed computing [Fischer 1985] [Loui 1987]

DC : Classic Distributed Algorithms

- Decision-making
  - Consensus : different algorithms
  - Randomized consensus algorithm [Aspnes 1990]
  - Chandra–Toueg consensus algorithm [Chandra 1996]
  - Paxos consensus algorithms family [wikipedia/ Paxos]

DC : Classic Distributed Algorithms

- Decision-making
  - Consensus: Paxos Consensus Protocol [Lamport 1998]



DC : Classic Distributed Algorithms

- Decision-making
  - Consensus: Paxos Consensus Protocol [Lamport 1998]
  - **Quorums:** Quorums express the safety properties of Paxos by ensuring at least some surviving processor retains knowledge of the results.

DC : Classic Distributed Algorithms

- Decision-making
  - Consensus: Paxos Consensus Protocol [Lamport 1998]
  - **Quorums:** Typically, a Quorum is any majority of participating Acceptors. For example, given the set of Acceptors {A,B,C,D}, a majority Quorum would be any three Acceptors: {A,B,C}, {A,C,D}, {A,B,D}, {B,C,D}. More generally, arbitrary positive weights can be assigned to Acceptors and a Quorum defined as any subset of Acceptors with the summary weight greater than half of the total weight of all Acceptors.

DC : Classic Distributed Algorithms

- Decision-making
  - Consensus: Paxos algorithms family
  - … according to conditions : different kind of failures of (participants: acceptors, etc.), multi-Paxos, cheap Paxos, fast-Paxos (conflicting or not), BFT-Paxos, virtual-Paxos, etc.

DC Principles : Outline

- Introduction
  - What is Distributed Computing ?
  - Why distributing ?
  - What are the problems in distributing ?
- From theory …
  - Distributed Algorithms: Byzantine problem
    - Synchronizers, Logical clocks, etc.
    - Graph colouring, Mutual exclusion, Consensus, Self-stabilization, etc.
  - Complexity

DC Complexity

- 5 aspects of distributed computing system complexity [Ranganathan 2007] :
  - Task-Structure Complexity,
  - Unpredictability,
  - Size Complexity,
  - Chaotic Complexity
  - and Algorithmic Complexity

DC Complexity

- Task-Structure Complexity
  - Cyclomatic complexity (CC) is:

CC = E - N + p

where E = the number of edges of the task graph

N = the number of nodes of the task graph

p = the number of connected components

DC Complexity

- Task-Structure Complexity
  - Cyclomatic complexity (CC) mesures the decision points

CC=9-9+1=1

CC=24-20+1=5

DC Complexity

- Unpredictability
  - Entropy H of the system

$$H = \sum_{i=1}^{k} p_i \log_2\left(\frac{1}{p_i}\right)$$

  - Higher the entropy of the system, the more difficult to predict

  - k different states with probabilities p1, p2, etc
  - log2(1/pi) the surprisal factor

DC Complexity

- Size Complexity
  - Traditionally, the size of a distributed system is measured by the number of nodes, devices, services, applications or other components.
  - In addition, a distributed system may have high cognitive complexity if users need to be aware of a large number of concepts in order to use the system. A concept is any logical item of knowledge defined or used by the system. A concept includes abstract notions like file-types, security policies, context information, device characteristics and QoS parameters. A large number of concepts contributes to greater difficulty in understanding the system as a whole.

DC Complexity

- Chaotic Complexity
  - Chaotic Complexity refers to the property of systems by which small variations in a certain part of the system can have large effects on overall system behaviour.
  - Chaotic complexity makes it difficult to understand systems. It often results from a lack of modular design and from a number of inter-dependencies between different parts of the distributed system.

DC Complexity

- Chaotic Complexity
  - An important factor contributing to chaotic complexity is coupling between different components.
    - Components are data coupled if they pass data through scalar or array parameters.
    - Components are control coupled if one passes a value that is used to control the internal logic of the other.
    - Components are common coupled if they refer to the same global data.
    - Components are content coupled if they access and change each other's internal data state or procedural state.

DC Complexity

- Chaotic Complexity : measuring coupling between components is fan in - fan out complexity
  - Count of the number of data flows into and out of a component plus the number of global data structures that the component updates.

Complexity = Length * $(\text{Fan-in} * \text{Fan-out})^2$

Length is any measure of length such as lines of code.

DC Complexity

- Cognitive Algorithmic Complexity : Halstead's measures [Halstead 1977]

n_1 = the number of distinct operators
n_2 = the number of distinct operands
N_1 = the total number of operators
N_2 = the total number of operands

DC Complexity

- Cognitive Algorithmic Complexity : Halstead's measures [Halstead 1977]

Program length (N) = $N_1 + N_2$

Program vocabulary (n) = $n_1 + n_2$

Program Volume (V) = $N * (\log_2 n)$. The program volume measures the information content of a program or the size of implementation of an algorithm.

DC Complexity

- Cognitive Algorithmic Complexity : Halstead's measures [Halstead 1977]

Difficulty (D) = $(n_1/2) * (N_2/n_2)$. The difficulty of a program is also related to the error-proneness of the program.

Effort (E) = D * V . The effort to implement or understand a program is proportional to the volume and to the difficulty level of the program.

DC Complexity

- Algorithmic Complexity
  - The traditional definition of the complexity of an algorithm is in terms of its time and space requirements or its relation to Turing machines or universal computers
  - A simple example is the use of an $O(n^2)$ algorithm for sorting (like insertion-sort or bubble-sort) as opposed to an O(n.log n) algorithm (like quick-sort).

DC Complexity

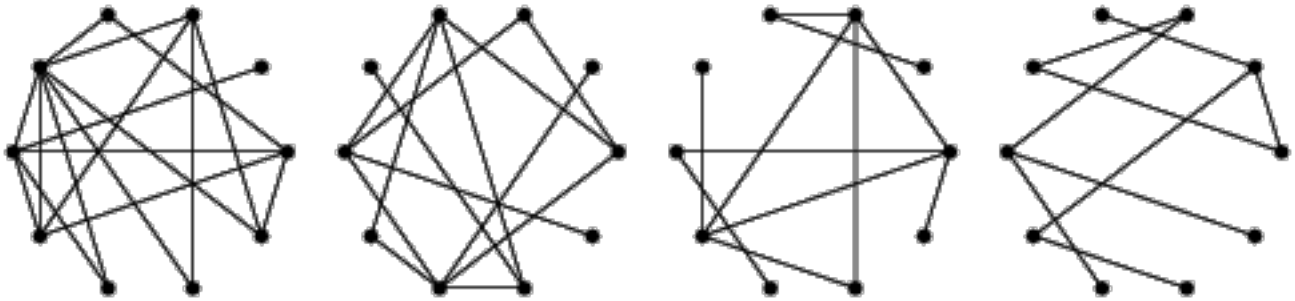
- Algorithmic Complexity
  - The traditional definition of the complexity of an algorithm is in terms of its time and space requirements or its relation to Turing machines or universal computers
  - A simple example is the use of an $O(n^2)$ algorithm for sorting (like insertion-sort or bubble-sort) as opposed to an O(n.log n) algorithm (like quick-sort).

DC Complexity

- Algorithmic Complexity
  - In distributed algorithms, another resource in addition to time and space is the number of computers
  - More attention is usually paid on communication operations than computational steps

DC Complexity

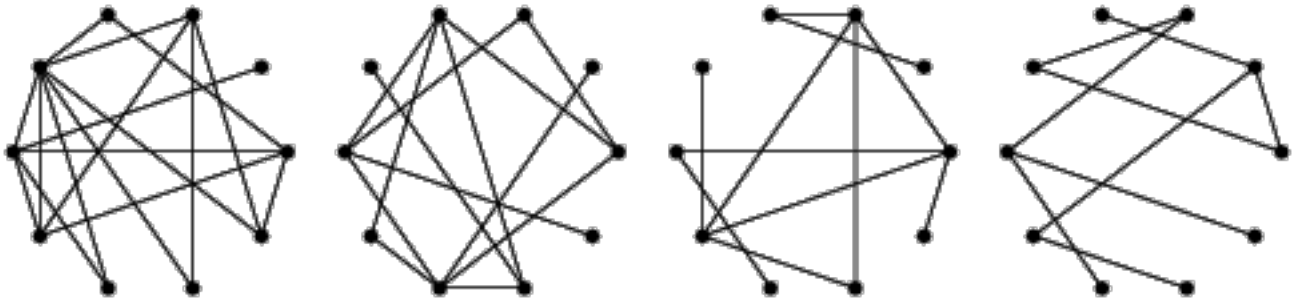
- Algorithmic Complexity
  - The complexity measure is closely related to the diameter D of the network
  - « longest shortest path »



D=3 D=4 D=5 D=7

DC Complexity

- Algorithmic Complexity
  - Diameter D of a graph : a graph's diameter is the largest number of vertices which must be traversed in order to travel from one vertex to another when paths which backtrack, detour, or loop are excluded from consideration



D=3 D=4 D=5 D=7

DC Complexity

- Algorithmic Complexity
  - The complexity measure is closely related to the diameter D of the network
  - Any computable problem can be solved trivially in a synchronous distributed system in approximately $2D$ communication rounds: simply gather all information in one location (D rounds), solve the problem, and inform each node about the solution (D rounds).

DC Complexity

- Algorithmic Complexity
  - The complexity measure is closely related to the diameter D of the network
  - Goal : if the running time of the algorithm is much smaller than *D* communication rounds, then the nodes in the network must produce their output without having the possibility to obtain information about distant parts of the network. In other words, the nodes must make globally consistent decisions based on information that is available in their *local neighbourhood*.

DC Principles : Outline

- … to practice
  - Architectures
  - Message Passing
  - Message-Oriented Middleware (MOM)

DC Principles : Outline

- … to practice
  - Architectures
  - Message Passing
  - Message-Oriented Middleware
- .. in the field of
  - Pervasive Computing ➔ Frédéric Le Mouël
  - Cloud Computing ➔ Julien Ponge
  - Grid Computing ➔ Yves Caniou

Bibliography

[Andrews 2000] Andrews, Gregory R. (2000), *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison–Wesley, ISBN 0-201-35752-6

[Awerbuch 1985] Baruch Awerbuch (1985), Complexity of Network Synchronization, Journal of the ACM, vol 32, n 4, p 804-823

[Eager 1989] Eager, D.L.; Zahorjan, J.; Lazowska, E.D. (1989), "Speedup versus efficiency in parallel systems", *Computers, IEEE Transactions on* , vol.38, no.3, pp.408-423, March, doi: 10.1109/12.21127

[Ghosh 2007] Ghosh, Sukumar (2007), *Distributed Systems – An Algorithmic Approach*, Chapman & Hall/CRC, ISBN 978-1-58488-564-1

[Hayashibara 2002] Naohiro Hayashibara, Adel Cherif, Takuya Katayama (2002), "Failure Detectors for Large-Scale Distributed Systems" , *21st IEEE Symposium on Reliable Distributed Systems (SRDS'02),* p 404

[Lamport 1998] Lamport, L. (1998). « The Part-time Parliament ». *ACM Transactions on Computer Systems*, 16(2):133-169. First appeared as DEC-SRC Research Report 49, 1989

[wikipedia/AL] *Algorithm*, http://en.wikipedia.org/wiki/Algorithm

[wikipedia/DA] *Distributed Algorithm*, http://en.wikipedia.org/wiki/Distributed_algorithm

[wikipedia/CP] *Computer Programming*, http://en.wikipedia.org/wiki/Computer_programming

[wikipedia/DC] *Distributed Computing*, http://en.wikipedia.org/wiki/Distributed_computing

Bibliography

[Cristian 1989] Cristian, F. (1989), “Probabilistic clock synchronization”, *Distributed Computing* 3(3):146-158.

[Gusella 1989] Gusella, R.; Zatti, S. (1989), "The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD", *Software Engineering, IEEE Transactions on* (IEEE) 15 (7): 847–853

[Ganeriwal 2003] Ganeriwal, S., Kumar, R., Srivastava, M. (2003). "Timing-Sync Protocol for Sensor Networks.", *The First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, p. 138-149

[Maroti 2004] Maroti, M., Kusy, B., Simon, G., Ledeczi, A. "The Flooding Synchronization Protocol.", *Proc. Of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[Marzullo 1984] K. A. Marzullo. Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service. Ph.D. dissertation, Stanford University, Department of Electrical Engineering, February 1984.

[Mills 2006] Mills, David L.. *Computer Network Time Synchronization: The Network Time Protocol*. Taylor & Francis / CRC Press. ISBN 0849358051

[Raynal 1986] Michel Raynal: *Algorithms for Mutual Exclusion*, MIT Press, ISBN 0-262-18119-3

Bibliography

[Akkoyunlu 1975] E. A. Akkoyunlu K. Ekanadham, R. V. Huber, Some constraints and tradeoffs in the design of network communications, Proceedings of the fifth ACM symposium on Operating systems principles SOSP'75

[Défago 2004] Défago, X., Schiper, A., and Urbán, P. 2004. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv*. 36, 4 (Dec. 2004)

[IEEE 1588] "IEEE 1588 Systems". National Institute of Standards and Technology (NIST)

[Lamport 1978] Lamport, L. (1978). "Time, clocks, and the ordering of events in a distributed system » *Communications of the ACM 21 (7): 558–565*

[Rentel 2005] Rentel, C.; Kunz, T. (March 2005), *A clock-sampling mutual network time-synchronization algorithm for wireless ad hoc networks, "A clock-sampling mutual network synchronization algorithm for wireless ad hoc networks", IEEE Wireless Communications and Networking Conference* (IEEE Press) **1**: 638–644, doi:10.1109/WCNC.2005.1424575 ISBN 0-7803-8966-2

[Skeen 1983] Skeen, Dale; Stonebraker, M. (May 1983). "A Formal Model of Crash Recovery in a Distributed System". *IEEE Transactions on Software Engineering* **9** (3): 219–228. doi:10.1109/TSE.1983.236608

[wikipedia/MA] *Marzullo's Algorithm* - http://en.wikipedia.org/wiki/Marzullo%27s_algorithm

[wikipedia/R] *Replication* - http://en.wikipedia.org/wiki/Replication_(computer_science)

Bibliography

[Chandra 1996] Chandra and Toueg. Unreliable failure detectors for reliable distributed systems. JACM 43(2):225–267, 1996.

[E.W. Dijkstra 2009] Cooperating Sequential Processes, manuscript, 1965, retrieved in 2009

[Schneider 2010] Schneider, J. (2010), "A new technique for distributed symmetry breaking", *Proceedings of the Symposium on Principles of Distributed Computing (SOSP)*

[Galager 1983] R. G. Gallager, P. A. Humblet, and P. M. Spira (January 1983). "A Distributed Algorithm for Minimum-Weight Spanning Trees". *ACM Transactions on Programming Languages and Systems* 5 (1): 66–77.

[Korach 1990] Ephraim Korach, Shay Kutten, Shlomo Moran (1990). "A Modular Technique for the Design of Efficient Distributed Leader Finding Algorithms". *ACM Transactions on Programming Languages and Systems* 12 (1): 84–101.

[Aspnes 1990] James Aspnes. Time- and space-efficient randomized consensus. *Journal of Algorithms* 14(3):414–431, May 1993. An earlier version appeared in *Ninth ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, August 1990, pp. 325–331.

[wikipedia/GC] *Graph coloring* - http://en.wikipedia.org/wiki/Graph_coloring

[wikipedia/LE] *Leader election* - http://en.wikipedia.org/wiki/Leader_election

Bibliography

[Fischer 1985] Fischer, Michael J. ; Nancy A. Lynch; Michael S. Paterson (April 1985). « Impossibility of Distributed Consensus with One Faulty Process » *Journal of the ACM* ***32*** *(2): 374–382*

[Guerraoui 2010] Rachid Guerraoui, Nikola Knezevic, Vivien Quéma, Marko Vukolic: The next 700 BFT protocols. EuroSys 2010: 363-376

[Loui 1987] Loui, M. C.; Abu-Amara, H. H. (1987). "Memory requirements for agreement among unreliable asynchronous processes". In Preparata, F. P.. *Advances in Computing Research*. **4**. Greenwich, Connecticut: JAI Press. pp. 163–183.

[Halstead 1977] Halstead, M. Elements of Software Science, Operating, and Programming Systems Series Volume 7, Elsevier, 1977.

[Ranganathan 2007] Anand Ranganathan, Roy H. Campbell « What is the complexity of a distributed computing system? » *Complexity*, Vol. 12, No. 6. (2007)

[wikipedia/Paxos] *Paxos Algorithm* - http://en.wikipedia.org/wiki/Paxos_algorithm