Introduction to Distributed Algorithms
Chapter 1: Models

Akka Zemmari

LaBRI - Université de Bordeaux

Content

Distributed Systems and Networks

Definitions [Tel]

- A distributed system is an interconnected collection of autonomous computers, processes, or processors. The computers, processes, or processors are referred to as the nodes of the distributed system.
- To be qualified as autonomous, the nodes must at least be equipped with their own private control,
- To be qualified as interconnected, the nodes must be able to exchange information.

Distributed Systems and Networks

Definitions [Tel]

- A distributed system is an interconnected collection of autonomous computers, processes, or processors. The computers, processes, or processors are referred to as the nodes of the distributed system.
- To be qualified as autonomous, the nodes must at least be equipped with their own private control,
- To be qualified as interconnected, the nodes must be able to exchange information.

Distributed Systems and Networks

Definitions [Tel]

- A distributed system is an interconnected collection of autonomous computers, processes, or processors. The computers, processes, or processors are referred to as the nodes of the distributed system.
- To be qualified as autonomous, the nodes must at least be equipped with their own private control,
- To be qualified as interconnected, the nodes must be able to exchange information.

Distributed Systems and Networks

Goals of a distributed system

The computers coordinate their activities and to share hardware and software and data, so that users perceive it as a single, integrated computing service with a well-defined goal.

Lamport:

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

Distributed Systems and Networks

Goals of a distributed system

The computers coordinate their activities and to share hardware and software and data, so that users perceive it as a single, integrated computing service with a well-defined goal.

Lamport:

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

Distributed Systems and Networks

Why distributed system

- Geographic distribution of processes
- Resource sharing (example: P2P networks, grids)
- Computation speed up (as in a grid or cloud)
- Fault tolerance

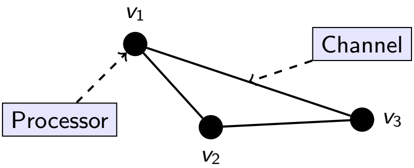
Distributed Systems and Networks

Some common problems

- Spanning tree
- Colouring problem
- Maximal Independent Set (MIS) problem
- Election problem
- Fault Tolerance
- Distributed snapshot

Network → graph

Simple (Un)oriented Graph

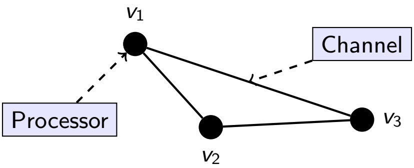


Notation

- The graph is denoted $G = (V, E)$, where V is the set of vertices (in graph theory, nodes are named vertices) and $E \subset V \times V$ the set of edges. In our example: $V = \{v_1, v_2, v_3\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$.
- The number of nodes in the network (the size of the network) is denoted n and the number of edges m.

Network → graph

Simple (Un)oriented Graph

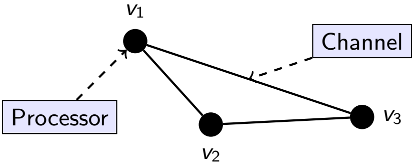


Notation

- The graph is denoted $G = (V, E)$, where V is the set of vertices (in graph theory, nodes are named vertices) and $E \subset V \times V$ the set of edges. In our example: $V = \{v_1, v_2, v_3\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$.
- The number of nodes in the network (the size of the network) is denoted n and the number of edges m.

Network $\rightarrow$ graph

Simple (Un)oriented Graph



Notation

- The graph is denoted $G = (V, E)$, where V is the set of vertices (in graph theory, nodes are named vertices) and $E \subset V \times V$ the set of edges. In our example: $V = \{v_1, v_2, v_3\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$.
- The number of nodes in the network (the size of the network) is denoted n and the number of edges m.

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  $\rightarrow$ communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  $\rightarrow$ communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  $\rightarrow$ communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  $\rightarrow$ communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  $\rightarrow$ communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  $\rightarrow$ communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  $\rightarrow$ communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  $\rightarrow$ communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  $\rightarrow$ communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  $\rightarrow$ communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

- Message passing communication: nodes exchange information by sending and receiving messages to and from their neighbours.
  → communications use instructions `send()/receive()`.
- Shared memory communication: nodes can exchange information by reading and writing data to the shared region.
  → communications use instructions `read()/write()`.
- Radio communication: when sent by a node, a message is received by all its neighbors.
- ...

In this talk, we consider message passing communication

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

Size of the messages

- $\mathcal{LOCAL}$ model: messages are of unbounded size (theoretical model).
- $\mathcal{CONGEST}$ model: messages are of size $O(\log n)$ bits.
- $O(1)$ messages model.
- Beeping model: nodes can only beep (and hence can hear beeps).
- ...

In this talk, we consider ALL this models

Communication Models

- Transmissions can be *simplex*, *half-duplex* or *full-duplex*
  $\rightarrow$ graphs are directed or undirected.
- Communication links are FIFO (First In First Out)
  $\rightarrow$ if a node sends two messages m_1 and m_2 in this order, message m_1 will be received before message m_2.

In this talk, we assume the transmissions are full-duplex and FIFO.

Communication Models

- Transmissions can be *simplex*, *half-duplex* or *full-duplex*
  $\rightarrow$ graphs are directed or undirected.
- Communication links are FIFO (First In First Out)
  $\rightarrow$ if a node sends two messages m_1 and m_2 in this order, message m_1 will be received before message m_2.

In this talk, we assume the transmissions are full-duplex and FIFO.

Synchronous/Asynchronous

Types of synchrony

- Synchronous clocks: Physical clocks are synchronized (Send and receive can be blocking or non-blocking).
- Synchronous processes: Lock-step synchrony.
- Synchronous channels: Bounded delay.
- Synchronous communication: Communication via handshaking.

Examples

- Postal communication is asynchronous.
- Telephone communication is synchronous.

Distributed Algorithms

Definitions [Tel]

A distributed algorithm for a collection $P = \{p_1, p_2, \cdots, p_n\}$ of processes is a collection of local algorithms, one for each process in P.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Analysis of Distributed Algorithms

Time Complexity [Peleg00]

- Round for each processor:
  1. Send messages to (some) the neighbors,
  2. Receive messages from (some) the neighbors,
  3. Perform some local computations.
- Time complexity: maximum possible number of rounds needed until every node has completed its computation.
- Message complexity: the total number of exchanged messages.

Termination of Distributed Algorithms

Different types of Termination

- Implicit termination: the algorithm has a finite execution and reaches a last configuration where each node of the system has the correct output. However, the nodes do not know that they reached the last states of the execution of this algorithm.

Termination of Distributed Algorithms

Different types of Termination

- Weak local termination: each node knows when it gets it final value (or its final state) but needs to continue sending messages to its neighbors.
- Local termination: each node knows its final value and can stop executing the algorithm.

Termination of Distributed Algorithms

Different types of Termination

- Global termination: at least a node knows that all the nodes finished computation.
- Strong global termination: a node knows the algorithm is terminated in all the system.

Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]

Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]

$\boxed{\text{ALGORITHM}}$

Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]

$$\boxed{\text{INPUT}} \longrightarrow \boxed{\text{ALGORITHM}}$$

Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]

$$\boxed{\text{INPUT}} \longrightarrow \boxed{\text{ALGORITHM}} \longrightarrow \boxed{\text{OUTPUT}}$$

Randomized Algorithms

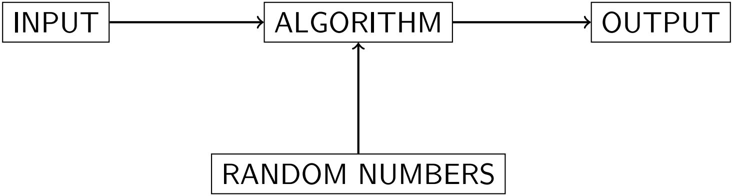
Randomized Algorithm [Motwani & Raghavan]

INPUT $\longrightarrow$ ALGORITHM $\longrightarrow$ OUTPUT

RANDOM NUMBERS

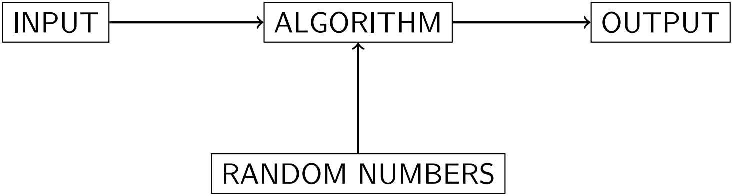
Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]



Randomized Algorithms

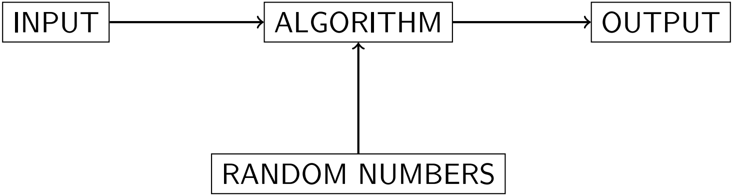
Randomized Algorithm [Motwani & Raghavan]



- In addition to input, algorithm takes a source of random numbers and makes random choices during execution,
- Behavior can vary even on a fixed input.

Randomized Algorithms

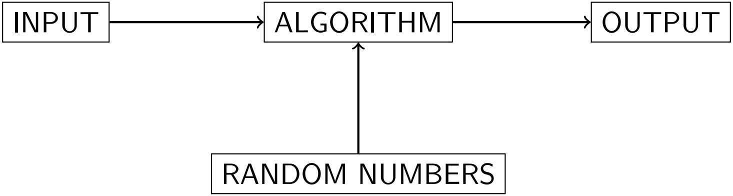
Randomized Algorithm [Motwani & Raghavan]



- In addition to input, algorithm takes a source of random numbers and makes random choices during execution,
- Behavior can vary even on a fixed input.

Randomized Algorithms

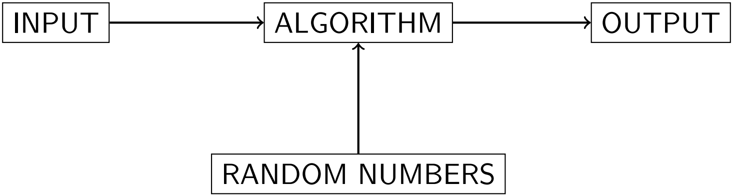
Randomized Algorithm [Motwani & Raghavan]



- In addition to input, algorithm takes a source of random numbers and makes random choices during execution,
- Behavior can vary even on a fixed input.

Randomized Algorithms

Randomized Algorithm [Motwani & Raghavan]



- Design algorithm $+$ analysis to show that this behavior is likely to be good, on every input.
  (The likelihood is over the random numbers only.)

Randomized Algorithms

Why randomized algorithms are useful?

(F. Magniez)

Compare a deterministic algorithm with runtime **1h**

1. to an algorithm with running time **1min** that
   1.1 aborts with probability 1/3, 1/2, 0.9
   1.2 fails with probability 1/3, 1/2, 0.9
2. to an algorithm that is always correct but whose running time is random and **1min** in average

Randomized Algorithms

Why randomized algorithms are useful?

(F. Magniez)

Compare a deterministic algorithm with runtime **1h**

1. to an algorithm with running time **1min** that
   1.1 aborts with probability 1/3, 1/2, 0.9
   1.2 fails with probability 1/3, 1/2, 0.9
2. to an algorithm that is always correct but whose running time is random and **1min** in average

Randomized Algorithms

Why randomized algorithms are useful?

(F. Magniez)

Compare a deterministic algorithm with runtime **1h**

1. to an algorithm with running time **1min** that
   1.1 aborts with probability 1/3, 1/2, 0.9
   1.2 fails with probability 1/3, 1/2, 0.9
2. to an algorithm that is always correct but whose running time is random and **1min** in average

Randomized Algorithms

Why randomized algorithms are useful?

(F. Magniez)

Compare a deterministic algorithm with runtime **1h**

1. to an algorithm with running time **1min** that
   1.1 aborts with probability $1/3$, $1/2$, 0.9
   1.2 fails with probability $1/3$, $1/2$, 0.9
2. to an algorithm that is always correct but whose running time is random and **1min** in average

Randomized Algorithms

Why randomized algorithms are useful?

(F. Magniez)

Compare a deterministic algorithm with runtime **1h**

1. to an algorithm with running time **1min** that
   1.1 aborts with probability $1/3$, $1/2$, 0.9
   1.2 fails with probability $1/3$, $1/2$, 0.9
2. to an algorithm that is always correct but whose running time is random and **1min** in average

Randomized Algorithms

Why randomized algorithms are useful? (In **distributed computing ...**)

- Assuming some hypotheses, many fundamental distributed tasks **can not be solved deterministically** (election, coloring, etc).
- In many cases, randomized algorithms can be more efficient than deterministic algorithms.

Randomized Algorithms

Why randomized algorithms are useful? (In **distributed computing ...**)

- Assuming some hypotheses, many fundamental distributed tasks **can not be solved deterministically** (election, coloring, etc).
- In many cases, randomized algorithms can be more efficient than deterministic algorithms.

Randomized Algorithms

Why randomized algorithms are useful? (In **distributed computing ...**)

- Assuming some hypotheses, many fundamental distributed tasks **can not be solved deterministically** (election, coloring, etc).
- In many cases, randomized algorithms can be more efficient than deterministic algorithms.

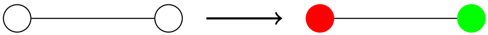
Randomized Algorithms

Definitions

- Las Vegas Algorithm: randomized algorithm which terminates with a positive probability and which produces a correct result.
- Monte Carlo Algorithm: randomized algorithm whose running time is deterministic, but whose output may be incorrect with a certain (typically small) probability.

Randomized Algorithms

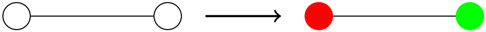
Example: Coloring an (anonymous) complete graph of size $n = 2$



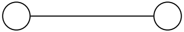
- Las Vegas:
- Monte Carlo:

Randomized Algorithms

Example: Coloring an (anonymous) complete graph of size $n = 2$



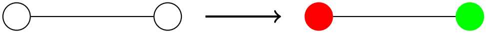
Monte Carlo



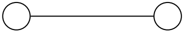
$color \leftarrow rand\{0, 1\};$ $\qquad$ $color \leftarrow rand\{0, 1\};$

Randomized Algorithms

Example: Coloring an (anonymous) complete graph of size $n = 2$



Las Vegas

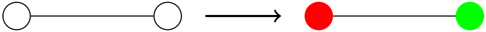


$color \leftarrow \emptyset$
repeart
  $b \leftarrow rand\{0, 1\}$;
  send(b);
  receive(b');
  if($b \neq b'$) then
    color $=$ b;
until color $\neq \emptyset$;

$color \leftarrow \emptyset$
repeart
  $b \leftarrow rand\{0, 1\}$;
  send(b);
  receive(b');
  if($b \neq b'$) then
    color $=$ b;
until color $\neq \emptyset$;

Randomized Algorithms

Example: Coloring an (anonymous) complete graph of size $n = 2$



Complexity? Correctness? of the two algorithms ...

Randomized Algorithms

Remark

Monte Carlo algorithm + Termination detection $\Rightarrow$ Las Vegas algorithm.

Bibliography I

H. Attiya and J. Welch.
Distributed Computing.
Wiley, 2004.

D. Peleg.
Distributed computing - A Locality-sensitive approach.
SIAM Monographs on discrete mathematics and applications, 2000.

G. Tel.
Introduction to distributed algorithms.
Cambridge University Press, 2000.

R. Motwani and P. Raghavan.
Randomized Algorithms.
Cambridge University Press, 1995.