Distributed Coin Tossing

François VERNADAT, Pierre AZEMA, Khalil DRIRA
LAAS-CNRS, 7 avenue du Colonel Roche 31077 Toulouse Cedex FRANCE

Abstract

This paper presents algorithms for distributed coin tossing. Distributed coin tossing furnishes a symmetrical way to break the symmetry in a distributed system. They allow the conflicts arising from competing remote sites to be overcome in a fair (equiprobable) manner: any site has an equal chance of being selected. It is also shown how these algorithms may be used to obtain a total order between competing sites of a system. Moreover, in each case the outputs of our algorithms are totally unpredictable .

1 Introduction

This paper presents algorithms for distributed coin tossing. Distributed coin tossing furnishes a symmetrical way to break the symmetry in a distributed system.

The proposed coin tossing mechanism may be used in many cases of distributed algorithms. It may be an alternative to other conflict resolution principles [RAY 88], e.g., static priority, timestamp messages [LAMP 78], ...

They allow the conflicts arising from competing remote sites to be overcome in a fair (equiprobable) manner: any site has an equal chance of being selected. Finally, we show how these algorithms may be used to obtain a total order between competing sites of a system.

The first algorithm selects a winner among n agents, while the second chooses it among only p competing agents, in a global configuration of n agents, $(p \leq n)$.

In the first case, the sites initially know that n nodes are actually competing. The problem lies in choosing one of the competing sites, then informing each one of the n sites about the winner of the round.

In the second case, competing sites do not know each other. The problem can be regarded as that of informing each one of n nodes about the p competing sites, then selecting one site among the p competing sites, and informing each one of the n sites about the winner of the round.

Finally we show how the previous algorithms may be used **without any additional communication** to obtain a total order between competing sites.

It is important to note, that our approach allows each site (competing or not) to have a complete knowledge of the ordering.

This common knowledge of the order is sufficient to determine useful classical hierarchies such as spanning tress [RAYM 89, AGR 89] but also more optimized structures such as balanced trees . This motivates the interest for deriving a complete order.

Many criteria may be considered for assessing distributed algorithms [RAY 88]: among these are the number of exchanged messages, the assumptions on the topology of the communication network, and on the quality of the data transfer service,...

Here, symmetry and unpredictability are the main criteria.

The unpredictable aspect of the output is precious when considering the problem of fault tolerance : in a case where the elected site is always (often) the same, it may be the privileged target of external aggressions and even in the absence of external assault its load will be more important increasing, its probability of failure.

The notion of symmetry appears more or less explicitly in many papers on distributed systems [BUR 81, BOU 87]: Symmetry can often be viewed as the absence of any kind of (hidden) monitor.

Luc Bougé defines the symmetry of an algorithm in the following manner [BOU 87]: "*An algorithm is symmetrical if during computation no process enjoys a statically defined privilege. If process P_u acquires some privilege during computation C, for instance by winning an election, then, for any process P_v there exists a computation C', during which P_v acquires this privilege. ... Symmetry does not prevent a process from knowing its identifier, but from using the latter to establish a priority among processes. Symmetry requires that any privilege be dynamically negotiated, the processes having the same rights and duties during this negotiation*".

The proposed algorithms are symmetrical because :

- each site runs the same code,
- no site benefits from a privilege known in advance,
- every site is equally involved in the final decision.

The level of symmetry of algorithms is clearly assessed in terms of probability, due to the specific aspect of the developed application: i.e., random choice. A simple combinatorial calculus shows that, whatever the site, its probability to win is not dependent on its identifier, or on its local choices.

As a result, the level of unpredictability may be also evaluated. In a normal case (in absence of cheating), the output of these algorithms is totally unpredictable. To predict the result of an election or to predict the total order associated with a consultation, a site needs to know all the values exchanged by its partners.

In terms of complexity, the algorithms require $(n-1)^2$ messages for n sites: each site sends a message to its $n-1$ partners, and waits for the respective acknowledgments. The size of the exchanged messages is bounded.
The presented algorithms for distributed coin tossing may be used in many fields of distributed computation. Solving the mutual exclusion problem is the most natural application; another concerns the resolution of conflicts by means of remote agreement.
This paper is organized as follows. Section 2 deals with the algorithm for randomly selecting one agent among n. Section 3 presents the algorithm for randomly selecting one agent from p among n. Section 4 presents an extension of the previous algorithms allowing to obtain a total order between competing sites.
Communication Assumptions:
The following hypotheses are made on the communication protocol for data transfer between sites.
(H1) Each site may (directly or indirectly) communicate with any one of its partners,
(H2) Transmission delays are bounded,
(H3) A reliable transmission medium is employed: messages are received in the order they were issued and are neither lost nor duplicated.
(H4) A node is identified by an identification number, from 1 to n, known to the other $n-1$ partners. This a priori numbering does not affect the symmetry of solutions as formally shown in the sequel. As a point of fact, for a particular node the probability to win does not depend on its identification number.

2 Random choice among n remote sites

When n sites initially compete, a solution involves the following two steps:
- break symmetry by selecting one site among n,
- let all other sites know the winner.

Each agent selects one value among the set {1,..,n} and sends it to each one of its $n-1$ partners. Then a site knows n values: its own choice and the $n-1$ choices of its partners. A site checks whether it is the winner or not by doing the following computation: a site wins iff the sum modulo n of these values plus one is equal to its identification number.
- Let $(a_1, \cdots, a_n)$, be the vector of exchanged values by n sites,
- Let G: $\{1,..,n\}^n \mapsto \{1,..,n\}$ be the mapping defined by

$$G(a_1, \cdots, a_n) = 1 + [(\sum_{j=1}^{j=n} a_j) \ \ mod\ (n)]$$

Site i which has n values $a_1, \cdots, a_n$ wins iff $G(a_1, \cdots, a_n) = i$
Algorithm Properties:

Termination of the algorithm in a consistent state:
- computation terminates,
- one and only one site wins, the others lose,
- all sites know the winner.

Equiprobability to win for any site:
- each node has an equal chance of winning,
- any value selected by a single node leads to the same opportunity of winning for this node.

Note: The absence of cheating is based upon the fact that a node sends its value **before** receiving any value from its partners.

2.1 Computation Termination

As transmission delays are bounded, the value exchange takes a finite amount of time. The reliability of the medium ensures that any site knows the same set of n values at the end of exchanges .
Function G remains the same for all sites and uses the same values, thus leading to the same result in all sites.
Each site is assumed to possess a unique identification number i, $i \in \{1, \cdots, n\}$, ensuring that one and only one site wins, (the mapping G is surjective).
Finally, each site can identify the winner because it knows all the partners.

2.2 Equal probability to win

Let $P(i)$ be the winning probability of site i, and $P(i,v)$ the probability that site i wins and that it chooses value v.
It is correct to consider in the sequel only the probability $P(i,v)$ instead of considering $P_v(i)$ the probability that i wins given that it chooses v. This holds because we suppose that : *the number selected by each node is uniformly distributed on the range* $1, \cdots, n$. This implies that the probability that site i chooses value v, $P_i(v)$, is constant: independent of i and v.
To demonstrate fairness (in its usual sense) the following properties have to be satisfied.

(P1) for a site, the probability to win does not depend on its identification number:
$\forall n > 1, \forall i, j \in \{1,..n\} : \ P(i) = P(j)$
(P2) for a site, the probability to win does not depend on the value it selects:
$\forall n \in N$, $\forall i \in \{1,..n\}, \forall v, w \in \{1,..n\}$: $P(i,v) = P(i,w)$
(P3) for site i, sending value v, the probability to win does not depend on i or on v:
$\forall n \in N$, $\forall i, j \in \{1,..n\}, \forall v, w \in \{1,..n\}$: $P(i,v) = P(j,w)$
Property(P1) deals with weak equiprobability : Each site *globally* has the same chance of winning.
Property(P2) ensures that there exists no value which increases the probability of winning for a specific site.
Property (P3) deals with strong equiprobability : any site is considered in the same manner; its chances of winning do not depend on identification numbers and the selected values.
The following holds because all sites operate with the same set of values :
(P3) $\Rightarrow$ (P2) $\Rightarrow$ (P1).

2.2.1 Strong equiprobability proof (P3)

Before sending a value, the chance of winning does not depend on i, the identification number, or on v, the selected value.
$\forall n \in N$, $\forall i, j \in \{1,..n\}, \forall v, w \in \{1,..n\} : P(i,v) = P(j,w)$
Let $C(i,v)$ be the set of arrangements in which site i chooses value v and wins.

To show that $P(i,v)$ is independent of i and v, it suffices to show that the number of elements, (Card: cardinal), of $C(i,v)$ only depends on n.
$P(i,v) = Card(C(i,v))/Card(A_n^n)$
where A_n^k is the set of k-tuples whose domain is $\{1,\cdots,n\}$.
$C(i,v)$ **counting**
In the case of n sites n values are requested to determine a winner. In other words during computation of a winner, starting from a (n-1)tuple, an ultimate nth value is requested to select any one site.
Let $C_{(a_3,\cdots,a_n)}(i,v)$ be the set of values $a \in \{1,..,n\}$ such that site i wins by selecting value v, the $n-2$ other sites have already chosen values $(a_3,\cdots,a_v)$ respectively.
$C_{(a_3,\cdots,a_v)}(i,v) = \{b \in \{1,..,n\} \mid G(b,v,a_3,\cdots,a_v) = i\}$
Let A_n^{n-2} be the set of (n-2-)tuples of values in $\{1,..,n\}$.
The following equalities hold:

$$C(i,v) = A_n^{n-2} \times C_{(a_3,\cdots,a_v)}(i,v)$$

$$Card(C(i,v)) = n^{n-2} \times Card(C_{(a_3,\cdots,a_v)}(i,v))$$

The value of $Card(C_{(a_3,\cdots,a_v)}(i,v))$ has to be computed.
Let b be the least integer such that $b \in C_{(a_3,\cdots,a_v)}(i,v)$ and

$$b = [(i-1) - (v + \sum_{k=3}^{k=n} a_k)] \ \ mod\ (n)$$

proposition: b is the unique element of $C_{(a_3,\cdots,a_v)}(i,v)$
- all the elements of $C_{(a_3,\cdots,a_v)}(i,v)$ are congruent modulo (n)
- As $C_{(a_3,\cdots,a_v)}(i,v) \subset \{1,..,n\}$, then $C_{(a_3,\cdots,a_v)}(i,v) = \{b\}$.

As $Card(C_{(a_3,\cdots,a_v)}(i,v))) = 1$, then $Card(C(i,v)) = n^{n-2} \times 1$, finally:

$$[P(i,v) = Card(C(i,v))/Card(A_n^n)] \Rightarrow P(i,v) = 1/n^2$$

and since $P_v(i) = \frac{P(i,v)}{P_i(v)}$ (from $P(A \text{ given } B) = \frac{P(A \text{ and } B)}{P(B)}$) we finally deduce

$$[P(i) = \sum_{v=1}^{v=n} P_v(i).P_i(v) = \sum_{v=1}^{v=n} P(i,v)] \Rightarrow P(i) = 1/n$$

3 Random choice between p out of n sites

The algorithm presented in the former section selects a winner among n competing sites. To use it, any one among the n sites is supposed to know all the other sites and whether they are competing.
In most cases, this knowledge is not available since:
- not all sites are competing,
- a competing site does not know its competitors.

The former algorithm selects one site out of n. To apply this algorithm would require organizing a poll round to allow the competing sites to know each other. Then any site knows the p competing ones, and a random choice of one out of p may proceed.
With respect to the number of exchanged messages, this solution requires $(n-1)^2$ messages for the consulting phase and $(p-1)^2$ messages for random choice.
In this section, a variant to the former algorithm is proposed, which only requires $(n-1)^2$ messages to select one site between p sites, in an n site configuration, $(p \leq n)$. The variant consists in merging consulting and selecting phases, so that the total number of messages remains limited to $(n-1)^2$.
To carry out the poll and selection phases, the following rules are adopted:
one site initiates a random choice by immediately sending its ballot.
When it receives this value, the site concerned regards it as a "consult request".Then, the site is free to either participate by returning his ballot or to opt out by sending a dedicated value. In the sequel, the dedicated value is zero.

3.1 Solution

Each agent, wishing to vote selects one value out of $\{1,..,n!\}$ and broadcasts it to the $n-1$ other sites. A site which opts out returns value 0.
As in the previous section, we suppose that *the number selected by each node is uniformly distributed on the range* $1,\cdots,n!$. This makes the probability that site i chooses value v is constant: independent of i and v.
Note: the $n!$ value is discussed later, (section 3.2.3).
For a given vector $(a_1,\cdots,a_n)$, two kinds of nodes are considered:
- active: an active node did not choose value zero.
$Active((a_1,\cdots,a_n)) = \{i \in \{1,n\}$ such that $a_i > 0\}$
- passive: a passive node chose value zero.
$Passive((a_1,\cdots,a_n)) = \{i \in \{1,n\}$ such that $a_i = 0\}$

Following this exchange, each site knows n values: that is, the one it chose and the values chosen by its $n-1$ partners.
The first step consists in renumbering the active sites from 1 to p: the idea is to associate with each active site its rank in the ballot, that is from 1 to the number of active sites p.

3.1.1 Relabeling Principle and Local Computation

Roughly speaking, the rank of site i, denoted N_E(i), will be equal to i minus the number of passive sites whose identifier is less than i.
Let $Passive((a_1,\cdots,a_n)) \cap \{1,\cdots i-1\}$ be the set of passive sites whose identification numbers are less than i. Then
N_E(i) = i - $Card([Passive((a_1,\cdots,a_n)) \cap \{1,\cdots,i-1\}])$
To know the outcome, a site does the following computation:
The site wins iff the sum modulo v of these n values is equal to its rank minus one, where v is the number of

active sites. Computation can be detailed as follows:

- $(a_1, \cdots, a_n)$: vector of values exchanged by the n sites.
- $v = Card(Active((a_1, \cdots, a_n)))$: number of active sites.
- $G : \{0, .., n!\}^n \mapsto \{1, .., v\}$, mapping such that

$$G(a_1, \cdots, a_n) = 1 + [(\sum_{j=1}^{j=n} a_j) \ \ mod\ (v)]$$

Site i, which knows n values $a_1, \cdots, a_n$ wins iff $G(a_1, \cdots, a_n) = N\_E(i)$

3.2 Computation Correctness

The algorithm terminates in a consistent state:

- at the most, one active site wins,
- the passive sites lose,
- every site knows the winner.

Strong equiprobability of winning per active site: Two active sites have an equal probability of winning whatever the values they respectively chose.

3.2.1 The algorithm terminates in a consistent state

The proof is analogous to the one of Section 2.1. Termination is again guaranteed and every site knows the same set of values, at the end of the round.

- As the computation function G remains the same irrespective of the site and uses the same set of values, then the output is identical on all sites.
- By construction, G is a surjective mapping from the set of choice vectors on the set $\{1, \cdots, Card(Active)\}$, the result selects a single active site, (i.e. whose rank $\in \{1, \cdots, Card(Active)\}$).
- Every site can compute the rank of any other site, and therefore the winner's identity.

3.2.2 Equiprobability of winning

As in Section 2.2.1, the strong equiprobability condition (P3) is directly shown.
For any configuration of p sites ($p \leq n$), the probability of winning must not depend on the identification number and on chosen values.
The following notations are used.

$C_{\uparrow<v_1, v_2, \cdots, v_v>}$, set of rounds where active sites are $v_1, v_2, \cdots, v_v$.

$C_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i)$, subset of $C_{\uparrow<v_1, v_2, \cdots, v_v>}$ for which site v_i ($v_i \in \{v_1, \cdots, v_v\}$) wins.

$C_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v)$, subset of $C_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i)$ for which site v_i wins by selecting value v.

$P_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i)$, site v_i probability of winning when competing with sites $v_1, v_2, \cdots, v_v$.

$P_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v)$, site v_i probability of winning when competing with sites $v_1, v_2, \cdots, v_v$ and when it chooses value v.

$C_{(a_3, \cdots, a_n)}(v_i, v)$, set of values $b \in \{1, .., n!\}$ such that $G(b, v, a_3, \cdots, a_n) = N\_E(v_i)$

Then $P_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v) = Card(C_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v))/Card(C_{\uparrow<v_1, v_2, \cdots, v_v>})$
where $C_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v) = A^{v-2}_{n!} \times C_{(a_3, \cdots, a_n)}(v_i, v)$
& $Card(C_{\uparrow<v_1, v_2, \cdots, v_v>}) = A^v_{n!}$
The last computation concerns $C_{(a_3, \cdots, a_n)}(v_i, v)$

$C_{(a_3, \cdots, a_n)}(v_i, v)$ counting
Let b the least integer such that $b \in C_{(a_3, \cdots, a_n)}(v_i, v)$ and

$$b = [(v_i - 1) - (v + \sum_{k=3}^{k=v} a_k)] \ \ mod\ (v)$$

Let (b_i) be a sequence of values derived from b by:
- $b_1 = b + v$
- $b_m = b + m \times v$

By construction, any value b_i less than $n!$ is included in $C_{(a_3, \cdots, a_n)}(v_i, v)$. Furthermore, any element of $C_{(a_3, \cdots, a_n)}(v_i, v)$ is congruent to b modulo v.
To compute $C_{(a_3, \cdots, a_n)}(v_i, v)$ it suffices to compute the greatest possible value m such that $b + m \times v < n!$.
In other words, find m such that $b + m \times v \leq n!$ and $b + (m+1) \times v > n!$
proposition: m = (n! / v) - 1

- As $v < n$, v divides $n!$, and m is indeed an integer
- $b + (m+1) \times v = b + (n!/v) \times v$ then $b + (m+1) \times v > n!$
- $b + m \times v = b + n! - v$ where $b \in \{1, ..v\}$ then $b + m \times v \leq n!$

Finally, we get

$$C_{(a_3, \cdots, a_n)}(v_i, v)) = \{b, b + v, \cdots, b + ((n!/v) - 1) \times v\}$$
$$Card(C_{(a_3, \cdots, a_n)}(v_i, v)) = n!/v$$

$$P_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i, v) = 1/(n! \times v) \quad \& \quad P_{\uparrow<v_1, v_2, \cdots, v_v>}(v_i) = 1/v$$

3.2.3 Choice of $n!$

To arrive at an equal weak probability of winning (P1) (cf 2.2) in the case of p active sites $v_1, v_2, \cdots, v_p$ must satisfy (cf notations Section 3.2.2)
$C_{\uparrow<v_1, v_2, \cdots, v_p>}(v_i) = C_{\uparrow<v_1, v_2, \cdots, v_p>}(v_j) \ \ \forall i, j \in \{1, .., p\}$
In addition

$$C_{\uparrow<v_1, v_2, \cdots, v_p>} = \bigcup_{i \in \{1, \cdots, p\}} C_{\uparrow<v_1, v_2, \cdots, v_p>}(v_i)$$

The following must hold: $Card(C_{\uparrow<v_1,v_2,\cdots,v_p>}) = p \times Card(C_{\uparrow<v_1,v_2,\cdots,v_p>}(v_i))$
In general, $Card(C_{\uparrow<v_1,v_2,\cdots,v_p>}) = V^p$ where V is the cardinal of the set of possible ballots, and p the number of active sites.
For p to divide V^p, V must be a multiple of p.
To ensure the weak equal probability whatever the number of active sites, V^k must be a multiple of k for all $k \leq n$. The least possible value of V is then the least common divisor of all integers less than or equal to n.
The strong equal probability (P3) is reached as soon as the number of elements of the set of possible values is a multiple of the least common divisor of $1, .., n$ (namely $n!$).

4 Obtaining a Total Order

In this section, we show how the previous algorithms may be used **without any additional communication** to obtain a total order between competing sites known by all the sites at the output of the algorithm. As previously, this order is totally unpredictable, and does not depend on the sites numbering.
The principle is quite simple, the total order is obtained by iterating the election process. The first site in the order is the elected site (in the sense of the previous sections), after that this site is considered as passive by the other active sites and a new election (using **the same exchanged messages**) is performed to obtain the second site of the order, and so on .
More formally, we show how the set of exchanged messages may be used to obtain a bijection, B, between *Active*, the set of the active sites, and $\{1, 2, \cdots, n\}$ (where $n = Card(Active)$). To conclude, it is sufficient to consider the canonical order associated with this bijection .

Lemma:
For a bijection $b : A \mapsto \{1, 2, \cdots, n\}$, the relation $<_b \subset A \times A$ defined by
$a_i <_b a_j$ iff $b(a_i) < b(a_j)$ is a strict total order on A

Construction of a bijection
Let B : $\{1, 2, \cdots, n\} \mapsto Active$ defined as follows:
$B(1) = S_i$ iff $G(v_1, v_2, \cdots, v_s) = N\_E(i)$ (cf section 3.1.1)
$B(k) = S_i$ iff $G(f_k(v_1), f_k(v_2), \cdots, f_k(v_s)) = N\_E(i)$ $(\forall k \in \{2, \cdots, n\})$
where the f_k (for $k \in \{2, \cdots, n\}$) are defined as follows :
$f_k(v_i) = 0$ iff $v_i = 0$ or $\exists p < k$ such that $B(p) = i$
$f_k(v_i) = v_i$ otherwise

proof:
Since $Card(Active) = Card(\{1, 2, \cdots, n\}$, it is sufficient to prove that B is an injection
Let $i \neq j$ (by example $i < j$) and $S_k = B(i)$
by construction of B and f,
For j such that $k < j \leq n$

$B(j) = S_p$
iff $G(f_j(v_1), f_j(v_2), \cdots, \mathbf{f_j(v_k)}, \cdots, f_j(v_s)) = N\_E(p)$
with $f_j(v_k) = 0$
As $f_j(v_k) = 0$, S_k may not be elected (cf section 3.2) and S_p is necessary different of S_k
Then $i \neq j \Rightarrow B(i) \neq B(j)$

Unpredictablity of the obtained order:
As in previous sections, the obtained order in unpredictable. The only way to cheat consists in waiting all the values exchanged by its partners to decide its own value.

5 Conclusion

This paper presents distributed algorithms for distributed coin tossing among remote agents. An extension allowing to obtain a total order on the competing sites has been also presented.
The solutions arrived at feature symmetry as main characteristic. These algorithms fairly and equiprobably solve conflict occurring between remote sites: every competing site has the same chances of winning, every site is similarly involved in the final decision.
Due to the specific aspect of the developed application: i.e., coin tossing, the level of symmetry of algorithms has been clearly assessed in terms of probability. A simple combinatorial calculus has shown that, whatever the site, its probability to win is not dependent on its identifier, or on its local choices.
In a normal case (in absence of cheating), the output of these algorithms is totally unpredictable. To be able to cheat, i.e., to predict the result of an election or to predict the total order associated with a consultation, a site needs to know all the values exchanged by its partners.
The unpredictable aspect of the output is precious when considering the problem of fault tolerance : in a case where the elected site is always (often) the same, it may be the privileged target of external aggressions and even in the absence of external assault its load will be more important, increasing its probability of failure.
In terms of complexity, the algorithms require $(n-1)^2$ messages for n sites: each site sends a message to its $n-1$ partners, and waits for the respective acknowledgments.
The proposed coin tossing mechanism may be used in many cases of distributed algorithms. It may be an alternative to other conflict resolution principles, e.g., static priority, timestamp messages [LAMP 78], ...
An example of immediate application is furnished by mutual exclusion algorithms leader election or random choice among remote agents .
The interest to determine a total order between competing sites is also obvious. It is important to note, that our approach allows each site (competing or not) to have a complete knowledge of the ordering.
This common knowledge of the order is sufficient to determine classical hierarchies such as spanning trees but also more optimized structures such as balanced trees.

References

[AGR 89] D. AGRAWALA, A. EL ABBADI

An efficient solution to the distributed mutual exclusion problem

Proc. 8th ACM Symposium on PODC, (1989), pp. 193-200

[BOU 87] L. BOUGÉ

Modularité et Symétrie pour les Systèmes Répartis; Application au langage CSP

Thèse de Doctorat d'Etat, Université Paris VII, 1987.

[BUR 81] J.E. BURNS *Symmetry in Systems of Asynchronous Processes*

Proc of 22nd Annual Symposium on Foundation of Computer Science, (1981), pp 169-174

[LAMP 78] L. LAMPORT

Time, clocks and the ordering of events in a distributed system

Comm. ACM, vol.21,7, (1978), pp. 558-565

[RAYM 89] K. RAYMOND

A tree-based algorithm for distributed mutual exclusion

ACM Trans on Comp Systems, Vol. 7. 1. (1989), pp 61-77

[RAY 88] M. RAYNAL

Networks and Distributed Computations: concepts, tools and algorithms

The MIT Press (1988)