General and efficient decentralized consensus protocols II

Jean-Claude Bermond, Jean-Claude König

GENERAL AND EFFICIENT DECENTRALIZED CONSENSUS PROTOCOLS II

Jean-Claude Bermond and Jean-Claude König[1]

Laboratoire de recherche en informatique
Unité Associée au C.N.R.S n^o 410
Bât 490, Université Paris-Sud
91405 ORSAY Cedex, FRANCE

In a preceding paper [6] we have given and analyzed a "consensus protocol". Such a protocol is a distributed algorithm which computes a function or predicate whose arguments are distributed on the nodes (or processors) of a network. Applications include for example finding the maximum of the identities of the processor (election problem) or computing a routing table. We give here more applications and show the importance of the network on which the algorithm is applied. In particular we show that on a de Bruijn network we have a D-phase algorithm with a number of messages roughly $2Dn^{1+\frac{1}{D}}$ (where D is the diameter and n the number of vertices of the network). This solves a conjecture of Lakshman and Agrawala [14]. If we restrict our attention to interconnection networks with a fixed maximum degree the de Bruijn or analogues networks appear to be well adapted with a running time of order $\log(n)$ and a message complexity of order $n\log(n)$. We also give refinements of the method which reduce the message complexity on these networks to $O(n)$.

1 Introduction.

In this paper we consider distributed algorithms called "consensus protocols" (see [17] and [18]). In such a protocol the initial data are distributed on the nodes (or processors) of the network. In order to compute the required result the nodes must cooperate and coordinate their activities by exchanging messages. At the end of the algorithm the result is either known by each node or distributed according the application. For example in an election the initial data is the identity of the node. When the algorithm is terminated each node must know the leader (for example the maximum of the identities). A second example is given by the computation of the sum of numbers. Initially each node contains one value and at the end it must know the sum of the values. Another important class of applications consists in finding information on the topology of the network. For example each node might want to know the diameter of the network or only its eccentricity (in that case the result are different in each node). The eccentricity of a node is the maximum of the distances between it and all the other nodes; the diameter is the maximum of the eccentricities. Each node might want to know a table of minimum routings (for example what channels it has to use to send messages to the other nodes). Here again the result is distributed (a node does not need to know the routing table of the others nodes). The word "consensus" indicates both the node agreement on the result and the rules of cooperation.

[1]This work has been done with the support of the french GRECO-PRC C^3.

In this paper we first recall a general algorithm given in [6]. We propose here a version for unidirectional networks (§2) and show how the protocol can be apply to compute the 2-connected component of a network. Then we study the complexity analysis and determine on what kind of networks it is very efficient (§3). If we suppose that the network is regular or has a fixed maximum degree we show that some families of networks like de Bruijn networks are well suitable. Using these networks we can answer a conjecture of Lakshman and Agrawala [14] and also obtained an algorithm with running time of order $\log(n)$ and a number of messages of order $n \log(n)$. A refinement of the method enables us to reduce the message complexity to $O(n)$ (§5). We also show how de Bruijn networks are well adapted to compute functions like the sum (§4).

Let us first precise our hypotheses and notation. (For more details see for example the books [17] or [18]). Nodes communicate only by exchanging messages on the channels of a given network. We assume that messages arrive in a finite time and unchanged. For time complexity analysis we will suppose that the transmission time on channel is bounded by some number τ. We do not need the "FIFO" assumption, that is we allow messages sent on a channel to arrive in a different order. A node can enter in the distributed protocol either spontaneously or after reception of a message concerning this protocol.

The network is supposed to be asynchronous. There is no central controller liked shared memory or global clock. Furthermore we want the cooperation control to be decentralized. In other words no node (or subset of nodes) can have a privileged behaviour. The protocol in a given node will depend only of the initial knowledge of the node and the result required. The algorithm in each node will be the same (symmetry properties see [5]). However each node has its own identity. Two different nodes have different identities. Initially each node knows only its identity, its initial data and the channels incident to itself and it can distinguish among them. No other topological information is known by the node. For example no node knows the total number of nodes. Such an hypothesis will allow modifications of the network without redefining all the network and the associated protocols.

The network will be modeled by a graph $G = (V, E)$ where the vertex set V represent the set of the nodes (or processors) and the edge set E the set of the channels. The channels can be either bidirectional, in that case the the graph is undirected, or unidirectional, in that case the graph is directed. The only slight difference in our algorithm is that in the case of bidirectional channels we send and receive messages on the channels incident to the processor and in the case of unidirectional channel we sent the message on the out-going channels and receive them on the in-going channels.

We have no hypothesis on the topology on the network except that it is connected (strongly connected for unidirectional channels) and that its topology does not change during the execution of the algorithm.

The number of nodes of the network will be denoted by n, the number of channels by m. The distance $d(x, y)$ from node x to node y is the length of a shortest path from x to y. Note that for a directed graph $d(x, y)$ might be different from $d(y, x)$. The in-eccentricity of a node $ecc(x)$ is the maximum of the distances from the other nodes to x: $ecc(x) = max_y d(y, x)$. The diameter D is the maximum distance between any couples of vertices.D is therefore the maximum of the eccentricities.

2 A general algorithm for consensus protocols.

We give here a slightly different version of the algorithm we proposed in [6]. The algorithm of [6] was given for bidirectional networks and was more complete concerning the termination and the filtering of the information. We refer the reader for a proof to [6]. Some of the concepts of the algorithm were developed by Awerbuch [2] and Gallagher [10] and are surveyed in the recent book of Raynal and Helary [18]. A similar algorithm has also been proposed by Tel [19] for unidirectional networks.

2.1 Algorithmic concepts.

The algorithm will consist of three classical parts

- initialization (INIT)
- several phases (PHASES*)
- termination (TERM).

The main idea is to use "exchange phases" to insure cooperation between the nodes. The use of phases induces a logical synchronization. During a given phase a node does successively three actions :

- send the new information on all its outgoing channels
- receive the information on all its incoming channels
- local computation, memorization and so on, in particular the node will compute the information he will send at the next phase.

Note that a node begins the phase $p+1$ only after receiving the informations sent by all its in-neighbors at phase p. But a given node can received on the same incoming channel messages sent by its neighbor at phase p, $p+1$ and so on. If the messages are received in the order they are sent we can for example suppose there is a message queue on each incoming channel and when all the queues contain a message deal with all the first messages. Otherwise we need to be able to reconize at what phase the message was sent. That can be done by adding to the message a label (or counter) indicating the number of the emission phase.

A node wakes up spontaneously or when its receives its first message. During the first phase it sends to its out-neighbors its identity and the local information. When he has received all the information of its in-neighbors he extracts the new information received to be sent at the next phase. It can be shown that at the end of the phase p a node P will have received the informations of all the nodes Q such that $d(Q,P) \leq p$. So the node P will have learned all the information at the end of the phase $ecc(P)$ (where $ecc(P)$ is the in-eccentricity of P). But it will discover this fact only at the end of the next phase where it will receive no new information. So the nodes do not terminate necessary at the same phase and therefore a node has to inform its out-neighbors that it has finished, for example by sending a special message end.

2.2 The algorithm.

The algorithm is given for an arbitrary node P_i

Variables :

- Inf_i is the global information known by P_i (at the beginning Inf_i consists of th identity of the node P_i plus its initial data).
- New_i is the new information obtained by P_i since the beginning of the current phase
- $Phase_i$ is the number of the phase P_i it is executing.
- R_i is the result obtained by P_i. It depends of the application (at the end R_i contain the required result).
- l indicates a channel (link) incoming in P_i.
- IN_i is the set of in-channels on which P_i can receive messages (when P_i receives a message "end" on some in-channel l it deletes this channel from IN_i). IN_i is initialized to the set of in-channels incident to P_i

Messages : A message contains a number of phases and the new information learnt by the processor in the preceding phase (or "end" if he has received no new information). To insure the termination the new information must contain at least the identities of the processors from which the processor learns new information. For example in a leader election the current maximum known by P_i can remain unchanged although the algorithm is not terminated.

Finally note that at $Phase_i$ the processor P_i consumes only the messages labelled with $Phase_i$ (the other messages are stored in a queue).

INIT

$New_i = Inf_i$

$Phase_i = 0$

PHASES

While $New_i \neq \emptyset$ do

begin

$Phase_i = Phase_i + 1$

For every out-channel do send $\prec Phase_i, New_i \succ$

$New_i = \emptyset$

For every in-channel l in IN_i do

begin

receive $\prec Phase_i, I \succ$ on l

If I="end"

then $IN_i = IN_i - \{l\}$

else begin

$New_i = New_i \cup (I - Inf_i)$

$Inf_i = Inf_i \cup I$

end

Compute R_i

```
        end
    end
TERM
    Phase_i = Phase_i + 1
    For every out-channel do send ≺ Phase_i, "end" ≻
    While IN_i ≠ ∅ do
    begin
        For every in-channel l in IN_i do
        begin
            receive ≺ Phase_i, I ≻ on l
            If I="end" then IN_i = IN_i − {l}
        end
        Phase_i = Phase_i + 1
    end
```

2.3 Application.

In [6] we have given different applications of the algorithm to solve election problem, computation of the routing table... Note that to compute some results (like the diameter or the routing table in the directed case) we might need to apply the algorithm twice. For example to compute the diameter we apply a first time algorithm to compute $ecc(P_i)$ which is given by $p-1$ where p is the value of $Phase_i$ when P_i discoves that $New_i = \emptyset$. Then we apply a second time the algorithm to find the maximum of the eccentricities. But in the second part we can save some time each processor starting the maximum finding algorithm when it has finished the first algorithm. The number of phases can be reduced by 1 (the processor knows the algorithm terminates at phase $ecc(P_i)$).

The protocol can also be used to determine the 2-connected components of an undirected network therefore answering a problem asked during the workshop in Amsterdam in July 1987. Note that a different kind of algorithm has been proposed by W. Hohberg [11] with a message complexity $O(m + n\log(n))$. As we will see after our algorithm has on good distributed networks the same complexity, but it is more simple and does not require an election.

In the undirected version of the algorithm it suffices to keep in P_i the edge on which we receive for the first time the identity of P_j. Suppose we receive this information at phase p on some edge a. If we receive again the identity of P_j on some other edge b at phase p or $p+1$ then a and b are in the same 2-connected component. Indeed suppose that P_i is an articulation vertex and that a and b are in different components, then at the end of the phase p P_i knows the identity of P_j via a, but the neighbor of P_i on the edge b cannot know P_j and cannot inform P_i neither at phase p or $p+1$. Conversely let us say that two edges incident to P_i satisfy the relation R if there exists a P_j such that the identity of P_j arrives first at phase p on a and then on b at phase p or $p+1$. Then it can be shown that an equivalence class of the transitive closure of R are exactly the edges of a 2-connected component incident to P_i.

So at the end of the algorithm we will have partitioned the edges incident to P_i into classes where two edges in the same class belong to the same 2-connected components. We therefore

know if P_i is a cut vertex or not and by sending the information through the network a second time we can find if needed the 2-connected components.

3 Complexity analysis and importance of the network topology.

3.1 Complexity analysis.

Number of messages : At each phase P_i sends $d^+(P_i)$ messages. As there are $ecc(P_i)+2$ phases for P_i, P_i send all together $(ecc(P_i)+2)d^+(P_i)$ messages. The total number of messages is therefore $\sum_i (ecc(P_i)+2)d^+(P_i)$.

Thus an upper bound is $(D+2)m$ where D is the diameter and m the number of arcs. In the undirected case if m denotes the number of edges and D the diameter the number of messages is bounded by $2(D+2)m$.

Bit complexity : Suppose that the initial data are of size $\log(n)$, then, as any information is transmitted at most once on a channel, the bit complexity is $O(nm\log(n))$.

Running time : Let τ be the maximum transmission delay on a channel. All the nodes will be woken up within time τD. Since each phase has an execution time bounded by τ the running time of the phase (and term) part is at most $(D+2)\tau$. So the running time of the algorithm is at most $2(D+1)\tau$.

3.2 What is the best topology?

The complexity analysis done above shows the importance of the network topology on the behaviour of the algorithm. If we want to minimize the running time we have interest to choose D as small as possible. If we want to minimize the bit complexity we have to choose m as small as possible. If we want to minimize the number of messages we have to minimize $(D+2)m$.

For example the best choice for the running time will be $D=1$, but in this case the network is complete and $m=n(n-1)$ or $\frac{n(n-1)}{2}$ according G is directed or not. The smallest value of m is obtained when G is a tree (or a symmetric directed tree) and in that case $m=n-1$. Among the trees there is one with diameter two, namely the star. This network appears to be a good one to apply the algorithm. But it is in fact a centralized network (a processor is joined to all the other processors) which does not satisfy our original requirement that no node has a privileged behaviour. Furthermore the networks of parallel architectures must satisfy the following technological constraint : the number of channels incident to a node must be small.

So in what follows we will consider networks with a maximum degree Δ (a maximum outdegree and indegree d in the directed case) bounded. In that case the number of edges (or arcs) is at most $m \leq \frac{\Delta n}{2}$ $(m \leq dn)$. The number of messages will be at most $(D+2)\Delta n$. In fact good topologies will have symmetry or uniformity properties and so we can assume that all the nodes have the same degree (or almost the same degree). In that case the number of messages is $(D+2)\Delta n$. So we have to consider the following kind of optimization problems : find networks of order n and maximum degree Δ having the smallest diameter

possible. Another approach considered by Lakshman and Agrawala [14] was to impose also uniformity constraints (regular network)and a given number of phases that is the diameter. In their paper they consider protocols with two phases and ask questions about k phases. In that case we can view the problem as finding networks with n vertices, diameter D and having the smallest Δ. Before to go further let us recall what is known on these problems and related ones.

3.3 (Δ, D)-graph problem.

A problem closely related to the ones mentioned above consists in finding the maximum number of vertices $n(\Delta, D)$ of a network with given maximum degree Δ and diameter D. (For more details see the survey [3] or the forthcoming book [1]). This number is bounded by the following function of Δ and D known as the Moore bound $n(\Delta, D) \leq 1 + \Delta + \Delta(\Delta-1) + \cdots + \Delta(\Delta-1)^{D-1}$. In the directed case a similar bound can be obtained where d is the maximum of the d^+ and d^- : $n(d, D) \leq 1 + d + d^2 + \cdots + d^D$.

It is known that these bounds are almost never attained. The best of the general known families are the de Bruijn networks or Kautz networks defined as follows.

The classical de Bruijn networks (named after the article of de Bruijn [7] although they were discovered earlier) can be defined using words on alphabets. The vertices are the words of length D constructed on an alphabet of d letters. The vertex $(x_1, x_2, \ldots, x_D)$ is joined by an arc to the vertices $(x_2, \ldots, x_D, \lambda)$ where λ is any letter of the alphabet. This digraph has indegree and outdegree d, d^D vertices and diameter D. To see that the diameter is D it suffices to consider the following simple routing to send a message from $(x_1, x_2, \ldots, x_D)$ to $(y_1, y_2, \ldots, y_D)$. The path is $(x_1, x_2, \ldots, x_D)$ $(x_2, \ldots, x_D, y_1)$ $(x_3, \ldots, x_D, y_1, y_2) \ldots (x_D, y_1, \ldots, y_{D-1})$ $(y_1, y_2, \ldots, y_D)$. This routing depends only on the address of the recipient of the message. Note that there exists an unique directed path of length D from any vertex x to any vertex y. The undirected associated graph is obtained by deleting the orientation. Then we have a graph of maximum degree $\Delta = 2d$ (the minimum degree is $2d-2$), diameter D and $\left(\frac{\Delta}{2}\right)^D$ vertices.

Kautz networks are defined in a similar way. The vertices are now words of length D constructed on an alphabet of $d+1$ letters, with the restriction that any two consecutive letters are different. The arcs are defined like on a de Bruijn network: there is an arc from $(x_1, x_2, \ldots, x_D)$ to $(x_2, \ldots, x_D, \lambda)$ with $\lambda \neq x_D$. In that case there exists a unique directed path of length D or $D-1$ from x to y (according $y_1 = x_D$ or $y_1 \neq x_D$). These digraphs have in- and out-degree d, diameter D and $d^D + d^{D-1}$ vertices.

Generalized de Bruijn and Kautz digraphs can be defined for any value of n by using integers modulo n. The vertices of these networks are the integers modulo n. In the case of generalized de Bruijn digraphs there exists an arc from i to j if and only if $j \equiv di + a$ with $0 \leq a \leq d-1$ and in the case of generalized Kautz digraphs there exists an arc from i to j if and only if $j \equiv -di - a$ with $1 \leq a \leq d$. It can easily be shown that for $n = d^D$ (resp $n = d^D + d^{D-1}$) we find the classical de Bruijn (resp Kautz) digraphs.

In the undirected case for small value of D or Δ one can construct better networks than de Bruijn or Kautz networks. For example using finite geometries one can construct for $D = 2$ (resp 3, 5) networks having D^2 (resp D^3, D^5) vertices (see [9]). For small Δ the best bounds are obtained by using random networks (see the article [4]). However Jerrum and Skyum [13] (see also Chung [8] for $\Delta = 3$) have constructed networks with a diameter

$D = 1.47 \log_2(n)$ for $\Delta = 3$; $D = 0.91 \log_2(n)$ for $\Delta = 4$; $D = 0.56 \log_2(n)$ for $\Delta = 6$ (De Bruijn and Kautz networks have a diameter $\log_2(n)$ for $\Delta = 4$ and $0.63 \log_2(n)$ for $\Delta = 6$).

3.4 Good networks.

Let us now apply the above result to the analysis of our protocol. For $D = 2$ our protocol is almost the same as the 2-phase protocol of Lakshman and Agrawala and requires a number of messages of order $(D+2)n\Delta \simeq 4n\sqrt{n}$. Similarly for $D = 3$ we have a 3-phase protocol with a number of messages $5n^{\frac{4}{3}}$. More generally on the de Bruijn or Kautz networks of diameter D our algorithm is a D-phase protocol using a number of messages of order $Dn^{1+\frac{1}{D}}$ in the unidirectional case and $2Dn^{1+\frac{1}{D}}$ in the bidirectional case. (indeed $n = d^D$ so $d = n^{\frac{1}{D}}$). This answers a problem asked by Lakshman and Agrawala [14]. They also solved independently the problem [15], but their protocols need strong symmetry properties and use a number of messages of order $D^2 n^{1+\frac{1}{D}}$. That comes from the fact that they use generalized hypercubes which are not as good as the de Bruijn or Kautz networks.

Lakshman and Agrawala fixed the number of phases or the running time or equivalently the diameter of the network. In fact, as we mentioned in 3.2 the design of the network might impose restrictions on the degree. In that case we can have a very good message complexity of order $n \log_2(n)$. It suffices to consider the directed de Bruijn network with $d^+ = d^- = 2$ or the associated undirected one with $\Delta = 4$, then $D = \log_2(n)$ and the number of messages is about $4n \log_2(n)$. Using the best of known networks of maximum degree 6 one can have a slightly better complexity of order $3.36 \log_2(n)$. Note that using generalized hypercube one obtains only a message complexity of order $n(\log_2(n))^2$ as $D = \Delta = \log_2(n)$.

From this analysis we can deduce different conclusions. Our protocols behave well on good networks of fixed degree, in particular on de Bruijn or Kautz networks (directed or not). So these networks appear to be well adapted to construct future distributed architectures. We will see that the use of some of their properties can reduce the computation of some functions. If we are on a network with a large degree, for example on a complete network, we have interest to simulate on it a de Bruijn or Kautz network to have a good message complexity or to reduce the quantity of information transmitted. One also can think to simulate random networks with a given small degree. By the result of Bollobás and de La Vega [4] these networks will almost surely have a good diameter of order $\log_2(n)$. This can work well if the links are randomly chosen in advance by some external person. An ideal solution would have been that each processor decides by itself to select at random a given number of outlinks to send the messages. But the problem is for a processor to know what is its sets IN_i that is on what links it is supposed to receive messages. To solve that problem there must be stronger hypotheses like all the processors wake up simultaneously (therefore after time τ the processors know their inlinks) or the processor knows a bound on the diameter of the network or some other topological information.

4 Computation of functions on de Bruijn networks.

As noted in §3.2 de Bruijn directed networks have the property that there exists an unique directed path of length D between any ordered pair of vertices. So if we know that the network is a de Bruijn directed network of diameter D (or a simulated one) we can apply the algorithm of §2.2 but without filtering or memorization. At each phase processors send all the information received at the preceding phase, but they do not keep it. Therefore any processor receives at the last phase all the information without redundancy. We gave here

the algorithm to compute the sum. Initially each processor contain a value $v(P_i)$ and at the end of the protocol we require that each processor contains the sum of values.

S_i will denote a variable containing the sum of the values received in the current phase.

Note that the same algorithm can be applied to compute any function of the $v(P_i)$.

Algorithm to compute the sum on a de Bruijn directed network :

$S_i = v(P_i)$

For $Phase_i = 1$ to D do

begin

send $\prec Phase_i, S_i \succ$ on all the out channels

$S_i = 0$

$Phase_i = Phase_i + 1$

For every in-channels l do

begin

receive $\prec Phase_i, I \succ$

$S_i = S_i + I$

end

end

At the end of the D^{th} phase S_i will be in each processor equal to the sum of the values $\sum_i v(P_i)$.

Proof : We show by induction on p that at the end of the phase p, $S_i = \sum_j \mu_p(P_j, P_i)v(P_j)$ where $\mu_p(P_j, P_i)$ denotes the number of directed path from P_j to P_i of length exactly p. (note that these paths can contain loops of repeated arcs). That is true for $p = 0$ or $p = 1$. Suppose that is true for some p. Then at the end of phase $p+1$, S_i is the sum of the values of S_k where P_k is an in-neighbor of P_i. By induction hypothesis $S_i = \sum_k \sum_j \mu_p(P_j, P_k)v(P_j)$ where P_k is an in-neighbor of P_i. As each path of length $p+1$ from P_j to P_i consists of a path of length p from P_j to P_k plus the arc P_kP_i where P_k is a predecessor of P_i we obtain $S_i = \sum_j \mu_{p+1}(P_j, P_i)v(P_j)$. At the end of the algorithm we therefore have $S_i = \sum_j \mu_D(P_j, P_i)v(P_j)$. But the existence of a unique directed path of length D from P_j to P_i gives $\mu_D(P_j, P_i) = 1$ for every P_j and so $S_i = \sum_j v(P_j)$.

Remark. The same algorithm can be applied on any network satisfying $\sum_{p=1}^{D} a_p A^p = J$ where A is the adjacency matrix associated to the graph G ($a_{ij} = 1$ if and only if there is an arc from P_i to P_j) and where J is the matrix having all its elements equal to 1. The above property can be expressed in terms of paths by noting that the coefficient a_{ij} of A^p represents the number of paths of length p from P_i to P_j. For example de Bruijn networks satisfy $A^D = J$ and Kautz networks $A^D + A^{D-1} = J$. Then the sum is given by R_i, where R_i is initialized to 0 and at the end of phase p, R_i is obtained by adding to the preceding value of R_i a_pS_i.

5 Variations and refinements.

Other ideas can be applied to reduce the number of messages using different notions of phases. For example, different elections algorithms has been designed on a ring which use phases, where an active processor sends messages to processors at some distance from it (either defined according some rule like in Hilschberg and Sinclair's algorithm or till the next active processors (see [16, chap.2])). However these algorithms use the ring topology and need to have active and passive processors. Here we propose a different approach which is very suitable for some classes of bidirectional networks.

At the end of the first phase each processor knows the identities of its neighbors. It decides to orient the edges from the largest identity to the smallest one. This orientation is well defines a directed graph without directed cycles.

Now during a current phase a processor will do two protocols. In a first part the messages will be sent according the orientation. A processor wait until it has received the information on all its incoming arcs and then send it on all its outgoing arcs. The processors which have no incoming arcs start immediatly the process. The processors which have no outgoing arcs start the second part by sending now messages on their incoming arcs. In the second part a processor wait until he has received all the messages on its outgoing arcs and send the information on its incoming arcs. The protocol terminates when a processor receives no new information in any past of the phase.

Now suppose that their exists a path between P_i and P_j with at most k changes of orientation then it can be shown that P_i will have learned the information at phase at most $2 + \lceil \frac{k}{2} \rceil$ So the parameter important for analysing the complexity of the algorithm is D^* which is the maximum over all pairs of processors P_i, P_j of $k(i,j)$ where $k(i,j)$ denotes the minimum number of changes of orientation on all the paths between P_i and P_j. Then the number of phases is at most $4 + \lceil \frac{D^*}{2} \rceil$ (one more phase to realize that there is no new information and one phase to inform its neighbors). The number of messages sent during a phase is $2m$. So the message complexity is at most $m(D^* + 8)$. The running time of the algorithm is at most $(D + 2 + \lambda D^*)\tau$ where λ is the length of the longest directed path in the directed graph ($D + 2$ corresponds to the waking up time plus the first and last phase).

Note that in all the networks D^* is less than D. In many cases it can be considerably smaller. For example in an hypercube or a de Bruijn network where the vertices are labelled in lexicographic order; $D^* = 1$. Furthermore every processor will know all the information at the end of the second phase. (the vertex $(0, 0, \ldots, 0)$ will know all the information at the end of the first part of this second phase). So the message complexity is of order of m. If we are on a de Bruijn network with fixed degree the message complexity is in $O(n)$. One can verify that the longest path according the orientation by lexicographic order is of order dl where d is the number of letter of the alphabet and so the running time is of order $\log(n)$ like in the original algorithm. Recently F. de La Vega (private communication) has shown that in a random graph of degree at least $O(\log(n))$ then $D^* = 3$ with a probability tending to one. So the expected message complexity will be of order $O(n \log(n))$ (to be compared with $O(n \log(n) loglog(n))$ before).

6 Conclusion.

We hope to have shown that the simple ideas of logical synchronization by phases give rise to general, simple and efficient protocols. The analysis of the complexity of the protocol

shows the importance of the network topology in particular of the diameter of the network. For example on distributed architectures like de Bruijn networks with a fixed maximum degree (4 for example) and a diameter of order $\log(n)$ we have algorithms with message complexity of order $n\log(n)$ and running time $\log(n)$. This fact and other examples show that networks like the de Bruijn ones are well suitable for distributed architectures. Finally we have given some refinements of the algorithm which reduce on these networks the number of messages to $O(n)$ while keeping the running time to $O(\log(n))$.

Acknowledgement : We thank G. Tel and M. Raynal for all their helpful comments concerning this article.

References

[1] D. Ameter and Max de Gree, *Graphs and Interconnections Networks*, forthcoming book.

[2] A. Awerbuch, Complexity of Network Synchronization, Journal of ACM, 32, 4, (1985), 804-823.

[3] J-C. Bermond, C Delorme, J-J. Quisquater, Strategies for Interconnection Networks: Some Methods from Graph Theory , Journal of Parallel and Distributed Computing, 3, (1986), 433-449.

[4] B. Bollobás, W. Fernandez de la Vega, The diameter of random regular graphs, Combinatorica 2, 2, (1982), 125-134.

[5] L. Bougé, Modularité et symétrie pour les Systèmes Répartis, Thèse d'état, Paris, March 1987.

[6] J-C. Bermond, J-C. König, M. Raynal, General and Efficient Decentralized Consensus Protocols, in *Distributed Algorithm, 2nd International Workshop*, Amsterdam 1987, Lectures Notes in Computer Science 312, Springer Verlag, (1988), 41-56.

[7] N.G. de Bruijn, A Combinatorial Problem, Koninklijke Nederlansle Akademie van Wetenschappen Proc., Ser. A49, (1946), 758-764.

[8] F.R.K. Chung, Diameter of graphs: old problems and new results, in *Proc. 18th South Eastern Conference on Combinatorics Graph Theory and Computing* 1987, Congressus Numerantium (1988).

[9] C. Delorme, Grands graphes de degré et diamètre donnés, Journal Européen de Combinatoire, 6, (1985), 291-302.

[10] R.G. Gallagher, Distributed minimum hop algorithms., MIT, Tech. Depart. LIDS-P-1175, (1982).

[11] W. Hohberg, How to Find Biconnected Components in Distributed Networks, Manuscript.

[12] J-C. König, Les réseaux d'interconnexion et les algorithmes distribués, Thèse, Orsay, April 1987.

[13] M. Jerrum, S. Skyum, Families of fixed degree graphs for processor interconnection, IEEE Trans. on Computers, vol. C-33, (1984), 190-194.

[14] T.V. Lakshman, A.K. Agrawala, Efficient Decentralized Consensus Protocols , IEEE Transactions on Software Engineering, vol. SE-12, no.5, (1986), 600-607.

[15] T.V. Lakshman, A.K. Agrawala, Communication Structure of Decentralized Commit Protocols, Proceedings The 6^{th} International Conference on Distributed Computing Systems, IEEE, Cambridge, Massachusetts, May 19-23, (1986), 100-107.

[16] M.Raynal, *Algorithmes distribués et protocoles*, Eyrolles, Paris, (1985).

[17] M. Raynal, *Systèmes Répartis et Réseaux, concepts outils et algorithmes*, Eyrolles, Paris, (1987).

[18] M. Raynal, J-M. Helary, *Synchronisation et contrôle des systèmes et des programmes répartis*, Eyrolles, Paris, (1988).

[19] G. Tel, Directed network protocols, in *Distributed Algorithm, 2^{nd} International Workshop*, Amsterdam 1987, Lectures Notes in Computer Science 312, Springer Verlag, (1988), 13-29.