Broadcasting and Gossiping in de Bruijn Networks

Jean-Claude Bermond, Pierre Fraigniaud

BROADCASTING AND GOSSIPING IN DE BRUIJN NETWORKS

JEAN-CLAUDE BERMOND[†] AND PIERRE FRAIGNIAUD[‡]

Abstract. Communication schemes based on *store and forward* routing, in which a processor can communicate simultaneously with all its neighbors (in parallel) are considered. Moreover, the authors assume that sending a message of length L from a node to a neighbor takes time $\beta + L\tau$. The authors give efficient *broadcasting* and *gossiping* protocols for the de Bruijn networks. To do this, *arc-disjoint spanning trees* of small depth rooted at a given vertex in de Bruijn digraphs are constructed.

Key words. broadcasting, gossiping, interconnection networks, de Bruijn graphs, disjoint spanning trees

1. Introduction. In the design and use of parallel computers, different elements are important. Among them are the topology of the interconnection network and the communication scheme. In this paper, we focus on communication problems. Important cases appearing in parallel algorithms are

Broadcasting: send a message from a given vertex to all the vertices, also called *OTA* (one-to-all).

Gossiping: send messages from all the vertices to all the other vertices, also called *ATA* (all-to-all) or total exchange.

An important literature on graph theory concerning this problem assumes that the communication cost is a constant and that only one port can be used by a processor at a given time (see the survey [15]). Here we consider the *store and forward* model, in which a vertex can simultaneously send and receive (eventually) *different* messages to and from all its neighbors. Indeed, this communication possibility corresponds to existing parallel computers (hypercubes and transputer-based machines). Moreover, the neighbor-to-neighbor communication time depends on a latency, or start-up time β, and on a data transfer time per element, or propagation time τ ($\frac{1}{\tau}$ is the bandwidth of a link). Sending a message of length L to a neighbor takes time $T = \beta + L\tau$ [20], [28], [31].

The hypothesis stating that processors are able to communicate simultaneously through all their ports is well known [20], [28], [31]. Many parallel computers satisfy this hypothesis. However, note that if the number of communication ports of a vertex is large, the start-up time (and in a less significant way, the propagation time) may grow with the number of ports used simultaneously [11]. We do not consider this problem in this paper. The reader is referred to [12] for a survey of broadcasting algorithms under several hypotheses. Note also that there exist other models of routing, such as *circuit-switched* or *wormhole* routing [8], [21]; however, in the case of intensive communications such as broadcasting or gossiping, they do not seem to offer any significative advantages over the *store and forward* model [30].

Clearly, the broadcasting and gossiping protocols depend on the topology of the interconnection network. The choice of this topology is critical in the design of parallel computers. Different goals can be pursued to increase the performance and minimize the cost. These goals can be expressed in terms of the graph (or digraph) that represents the interconnection network. The vertices of the graph correspond to processors, and the edges (or arcs) correspond to communication links between processors.

[†]I3S, Centre National de la Recherche Scientifique, Bât.4, Avenue Albert Einstein, Sophia Antipolis, 06560 Valbonne, France. This author's research was supported by the research program C3.

[‡]LIP-IMAG, Centre National de la Recherche Scientifique, Ecole Normale Supérieure de Lyon, 46, Allée d'Italie, 69364 Lyon Cedex 07, France. This author's research was supported by the research programs C3 and ANM, and by the Direction des Recherches et Etudes Techniques.

An important constraint claimed by many authors [7], [8], [17] is that very large-scale integration (VLSI) computing systems are wire limited, which corresponds to a fixed small degree in the associated graph. Furthermore, in the model of store and forward routing, the total transmission time depends on the diameter, which should be as small as possible. Thus, it is important to have graphs with a small diameter and a fixed degree. This is not the case for the hypercube, since the degree and diameter increase as the logarithm of the number of processors, or for the multidimensional meshes or torus, since they have diameters that are too large.

Two well-known topologies are the Kautz and de Bruijn networks, which have many interesting properties [6], [29]. In particular, these networks interconnect considerably more processors than the usual topologies, and they have small diameters and small and fixed degrees. Therefore the aim of this paper is to find efficient broadcasting and gossiping protocols in the de Bruijn digraph. Classical broadcasting in de Bruijn digraphs using the assumption of a constant communication time and the possibility of sending a message to at most one neighbor at any time has been considered in [5], [16].

This paper is organized as follows. In §2 we recall some definitions about directed graphs. In §3 we compute lower bounds and explain possible communication protocols for broadcasting in a given network. We show that an efficient way uses the construction of arc-disjoint spanning trees. Section 4 describes the de Bruijn graphs and digraphs and introduces some notation. Section 5 proposes arc-disjoint spanning trees of the de Bruijn digraphs, which can be used to perform asymptotically optimal broadcasting. In §6 a fast, greedy, gossiping algorithm is given. Section 7 explains how to translate the proposed algorithms from the de Bruijn digraphs to the undirected de Bruijn graphs. Finally, §8 concludes the paper.

2. Notation. A network of n processors is usually modeled by a graph or a digraph $G = (V, E)$ of order n. Three kinds of communication links are usually used.

1. Monodirectional links: messages can only be transferred in one direction. The network is modeled by a digraph.

2. Bidirectional *half duplex* links: such links can be used at a given time in at most one direction. The network is modeled by a graph.

3. Bidirectional *full duplex* links: each link can be simultaneously used in both directions. The network is modeled by a *symmetric* digraph.

In this paper we will consider digraphs only (symmetric or not), either corresponding to networks with monodirectional links, or networks with bidirectional *full duplex* links. In a digraph there is an arc from a node u to a node v only if u is able to send a message directly to v.

Let $d(u, v)$ be the distance between the vertices u and v, that is, the length of a shortest path from u to v. We will use $ecc(u)$ to denote the eccentricity of the vertex u, that is, $\max_{v \in V} d(u, v)$, and use D for the diameter of the digraph, that is, $\max_{u \in V} ecc(u)$. We will use "shortest-paths spanning tree rooted at r" to denote breadth first search tree rooted at r. In such a tree, the path from r to any other vertex v is a shortest path. Note that the depth of such a tree is $ecc(r)$.

Let $d^+(u)$ (resp. $d^-(u)$) denote the out-degree (or in-degree) of the vertex u, that is, the number of outgoing (or incoming) arcs from (or to) u. If the digraph is regular then all the vertices have the same out- and in-degree d. Otherwise $d^+_{\max}$ denotes the maximum out-degree over all the vertices (similar definitions are derived for $d^-_{\max}$, $d^+_{\min}$, and $d^-_{\min}$). Let $m^+(S, V - S)$ be the number of arcs going from S to $V - S$ and let $c_G(r) = \min_{S \neq V | r \in S} m^+(S, V - S)$. $c_G(r)$ can be regarded as the minimum number of arcs that must be deleted to make at least one vertex not reachable from r. Another interpretation of $c_G(r)$ is that there exist $c_G(r)$ arc-disjoint paths from r to any vertex of G. Moreover, $c_G(r)$ is the maximum number of

such paths that are arc disjoint (Menger's theorem). Let λ be the arc connectivity of G, that is $\lambda = \min_{r \in V} c_G(r)$. Note that $d^+(r) \geq c_G(r) \geq \lambda$.

Finally, we will use $b_G(r)$ to denote the minimum time of any broadcast initiated by a vertex r of G, and use g_G for the minimum time of any gossiping on G.

3. Broadcasting in networks.

3.1. Lower bounds. Recall that the communication time to send a message of length L from a node to a neighbor is of the form $\beta + L\tau$. Following [20], [31], we can obtain two different lower bounds by considering the total start-up time or the total data transfer time. First the broadcasting time $b_G(r)$ is at least $ecc(r)\beta$. Consider now a subset S of V containing r such that $m^+(S, V - S) = c_G(r)$, and let u be a vertex of $V - S$. The total bandwidth of the arcs between S and $V - S$ is $c_G(r)/\tau$ and therefore the minimum time to send the message from r to u is at least $(L/c_G(r))\tau$.

PROPOSITION 3.1. *In a digraph G, the time to broadcast a message of length L from a node r is at least* $\max(ecc(r)\beta, (L/c_G(r))\tau)$.

For example, in a D-cube used in full duplex mode, $c_G(r) = d^+(r) = d^-(r) = D$, and $ecc(r) = D$. We obtain $b_{D-\text{cube}}(r) \geq \max(D\beta, \frac{L}{D}\tau)$, which is the bound proposed by Johnsson and Ho in [20].

3.2. Communication protocols. There exist different ways to perform broadcasting from an originator r. The efficiency of these protocols depends on the ratio $\frac{\beta}{L\tau}$. The first protocol simply uses a shortest-paths spanning tree. The broadcasting time is $ecc(r)(\beta + L\tau)$ because there exists at least one vertex at distance $ecc(r)$ from r. If $L\tau \ll \beta$, this time is of the order $\beta ecc(r)$ approaching the lower bound and so we cannot do better.

In contrast, if $L\tau \gg \beta$, this time is of the order $L\tau ecc(r)$, which is far from the lower bound $L\tau/c_G(r)$. In case of long messages, we can improve the total time by cutting the messages in smaller packets. A classical technique is pipelining. Suppose we cut the message into $\frac{L}{B}$ packets of length B and send the packets one after each other on a shortest-paths spanning tree. The broadcasting time is $ecc(r)(\beta + B\tau)$ for the first packet to reach a farthest node, plus $(\frac{L}{B} - 1)(\beta + \beta\tau)$ for the other $\frac{L}{B} - 1$ packets following the first one to reach a farthest node. Hence the total time is $(\beta + B\tau)(ecc(r) + \frac{L}{B} - 1)$. This time is optimized by choosing $B_{\min} = \sqrt{L\beta/(ecc(r) - 1)\tau}$, its value being $(\sqrt{L\tau} + \sqrt{(ecc(r) - 1)\beta})^2$. If $L\tau$ is large compared with β, we have now a time of the order $L\tau$. Thus we have earned a factor of $ecc(r)$, but are still far from the lower bound.

Another technique consists of finding p spanning trees rooted at r and pairwise arc disjoint. We cut the message into blocks, each of size $\frac{L}{p}$, and send each block on a different spanning tree. Suppose the maximum depth of the spanning trees is h, then the broadcasting time is $h(\beta + \frac{L}{p}\tau)$. This technique has been used for networks such as hypercubes [20], folded hypercubes [18], star graphs, and k-ary hypercubes [25]. See [12] for a survey. Moreover, we can use the following theorem of graph theory by Edmonds [10] (see Lovász [24] for a short proof).

THEOREM 3.2 (Edmonds). *The maximum number of pairwise arc-disjoint spanning trees rooted at a vertex r is equal to $c_G(r)$.*

We can use these $c_G(r)$ spanning trees to obtain a broadcasting time $h(\beta + (L/c_G(r))\tau)$. Here again, if $L\tau \gg \beta$, we have a time of order $(hL/c_G(r))\tau$ saving a factor of $c_G(r)$ when h is close to $ecc(r)$.

Finally, we can combine the two techniques of pipelining and arc-disjoint spanning trees. With packets of size B, we obtain a time $h(\beta + B\tau)$ so that the first packet reaches a farthest node in the deepest spanning tree, plus $(L/(c_G(r)B) - 1)(\beta + B\tau)$ for the other $L/(c_G(r)/B) - 1$

packets following the first one to reach this farthest node. Hence the total time is $(\beta + B\tau)(h + L/(c_G(r)B) - 1)$. An optimal choice of the size B of the packets gives a broadcasting time $(\sqrt{L\tau/c_G(r)} + \sqrt{(h-1)\beta})^2$.

THEOREM 3.3. *If the maximum depth of $c_G(r)$ arc disjoint spanning trees rooted at a vertex r is h, then there exists a protocol of broadcasting from r whose time is $(\sqrt{L\tau/c_G(r)} + \sqrt{(h-1)\beta})^2$.*

If $L\tau \gg \beta$, we have a time of order $L\tau/c_G(r)$, matching the lower bound. If β and $L\tau$ are of the same order of magnitude, we have to balance the effects of the number of spanning trees and the depth of these trees, respectively. Ideally we want to find many spanning trees ($c_G(r)$ if possible) of maximum depth as small as possible. In the case of regular classical networks, $c_G(r) = \lambda = d$ and $ecc(r) = D$, and the best we can hope to find is d arc-disjoint spanning trees with a depth of at most $D + 1$. Our aim will be to construct arc-disjoint spanning trees of the de Bruijn networks. Note that for the two-dimensional torus, this problem has been completely solved [26].

Let us finish this section with some remarks.

1. Instead of using arc-disjoint spanning trees, we could use time-disjoint diffusion trees (meaning that at a given time, one arc is used in at most one diffusion tree). However, these trees are more difficult to find and do not allow pipelining.
2. There exists algorithms for finding arc-disjoint spanning trees of any given digraph, but the complexities of these algorithms are high, and there is no good bound on the depth of the trees obtained. So it is interesting to give explicit constructions.
3. It is possible to do the same study for the *half duplex* communication mode. See [14], [22], [27] for results on edge-disjoint spanning trees of an undirected graph.

4. de Bruijn digraphs. The de Bruijn digraph $B(d, D)$ [9] is a digraph in which the vertices are the words of length D on an alphabet of d letters (for instance $\{0, \ldots, d-1\}$). There is an arc from a vertex $x = (x_1, \ldots, x_D)$ to a vertex y if the $D-1$ first letters of y are equal to the $D-1$ last letters of x. That is, there is an arc from $x = (x_1, \ldots, x_D)$ to all the vertices $(x_2, \ldots, x_D, \alpha)$, $\alpha \in \{0, \ldots, d-1)\}$. Therefore, each node of $B(d, D)$ has an out and in degree d and it is easy to check that the diameter is D, and for every vertex r of $B(d, D)$, $c_G(r) = \lambda = d - 1$. The number of vertices is $n = d^D$. Figure 1 shows $B(2, D)$ with $D = 1, 2, 3$.

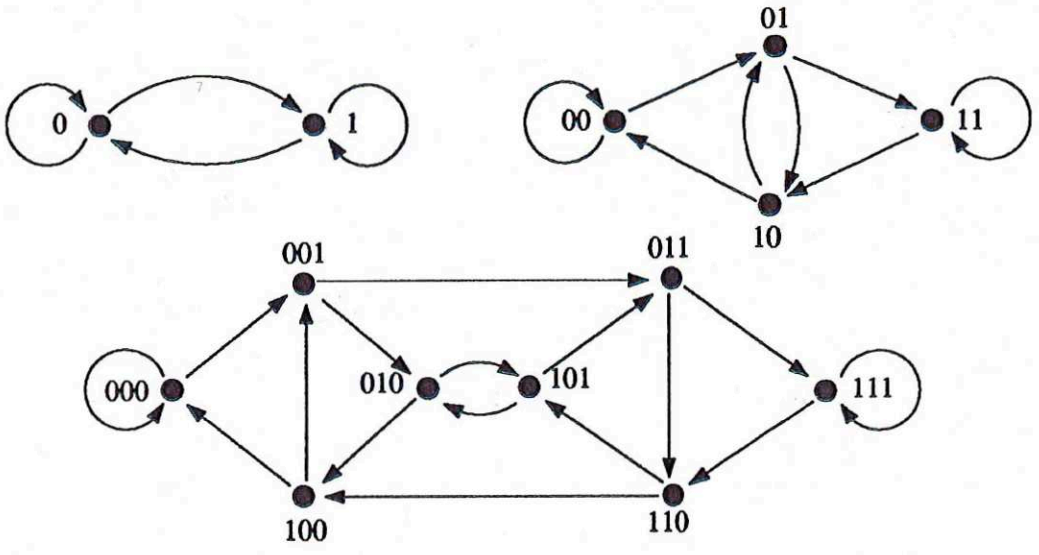

FIG. 1. $B(2, 1)$, $B(2, 2)$, *and* $B(2, 3)$.

One can note that, for a fixed degree d, the number of vertices of these digraphs is of the same asymptotic order as the directed Moore bound (which is equal to $d^D + d^{D-1} + \cdots + d^2 + d + 1$). Another important remark is that the degree and the diameter of these digraphs are independent and adjustable parameters.

The corresponding undirected graphs $UB(d, D)$ are built by omitting the direction of the arcs.

Remark. If we do not consider the loops, all the vertices $(\alpha, \alpha, \ldots, \alpha, \alpha)$ of $B(d, D)$ have in- and out-degree $d - 1$. Many solutions are proposed in the literature [2] to modify the de Bruijn graphs to make them regular, but we do not consider them in this paper. Indeed, links can be added to nodes of degree strictly less than the maximum degree to communicate with external devices such as front-end processors and peripherals.

In the following section, we use the notations $x = (x_1, \ldots, x_D)$ for a vertex of $B(d, D)$ and $e = [x_1, \ldots, x_D, \alpha]$ for the arc $(x_1, \ldots, x_D) \rightarrowtail (x_1, \ldots, x_D, \alpha)$. The capital letter W will represent any sequence of letters, and $|W|$ the length of this sequence.

5. Arc-disjoint spanning trees of the de Bruijn digraph. The main purpose of this section is to construct for a given vertex r of a de Bruijn digraph a set of $c_G(r) = d - 1$ arc-disjoint spanning trees all rooted at this vertex.

We will extensively use the *shortest-paths spanning tree* (or breadth first search tree). The shortest directed path from a vertex x to a vertex y is obtained by determining the longest sequence, common to the end of x and to the beginning of y. Suppose that this longest sequence is of length $D - h$ and of the form $z_1, \ldots, z_{D-h}$ and let $x = (x_1, \ldots, x_h, z_1, \ldots, z_{D-h})$ and $y = (z_1, \ldots, z_{D-h}, y_1, \ldots, y_h)$. Then the distance between x and y is h, and the unique shortest path between x and y contains successively the vertices

$$
\begin{aligned}
&x^0 = x \\
&x^1 = (x_2, \ldots, x_h, z_1, \ldots, z_{D-h}, y_1) \\
&\cdots \\
&x^i = (x_{i+1}, \ldots, x_h, z_1, \ldots, z_{D-h}, y_1, \ldots, y_i),\ 1 < i < h \\
&\cdots \\
&x^h = y.
\end{aligned} \tag{1}
$$

Figure 2 shows the two shortest-paths spanning trees rooted respectively at (000) and (111) in $B(2, 3)$. One can see that they are arc disjoint.

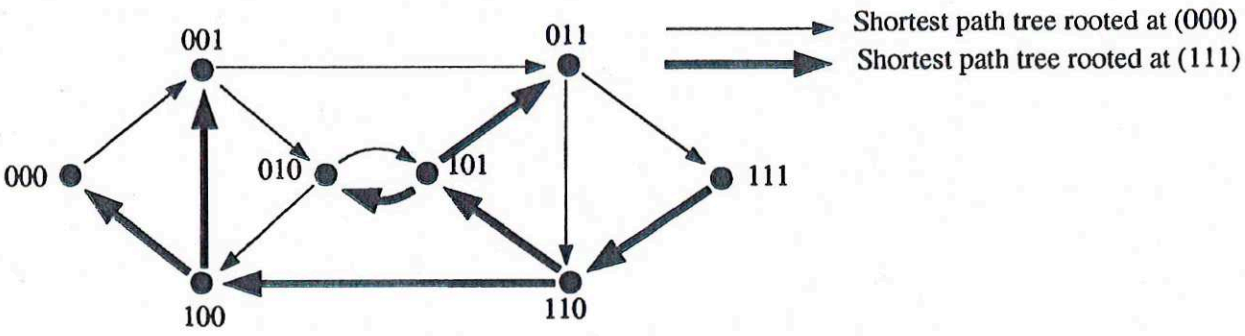

FIG. 2. *Two shortest-paths spanning trees in* $B(2, 3)$.

PROPOSITION 5.1. *The d shortest-paths spanning trees of the de Bruijn digraph, rooted respectively at the vertices $(\alpha, \ldots, \alpha)$, $\alpha = 0, \ldots, d-1$, are pairwise arc disjoint.*

Proof. Two arcs of different trees are of the form $[\alpha, \ldots, \alpha, W_1]$ and $[\beta, \ldots, \beta, W_2]$, with $|W_i| \leq D$, $i = 1, 2$. Hence, they are disjoint if $\alpha \neq \beta$. □

Thus, assuming that each of these nodes is connected to an external device, it is possible to load data from the front end to the parallel multiprocessor using d arc-disjoint spanning trees, with a bandwidth d/τ. This is a main point: de Bruijn graphs ensure high bandwidth between any external device and the multiprocessor. We show in the following how to ensure high bandwidth communications within the network.

It is more difficult to find arc-disjoint spanning trees of small depth all rooted at the same vertex in a de Bruijn digraph. It might be worth noting at this point that Imase et al. [19] have built $d-1$ arc-disjoint *paths* of length $\leq D+1$ between any two nodes to study the vulnerability of the de Bruijn digraph. Unfortunately, their construction cannot be used to find arc-disjoint spanning trees rooted at the same vertex.

Finally, let us note that we cannot construct more than $d-1$ arc-disjoint spanning trees rooted at a given vertex due to the presence of loops at the vertices $(\alpha, \ldots, \alpha)$, $\alpha = 0, \ldots, d-1 (c_G(r) = \lambda = d-1)$.

First we present a simple construction of $d-1$ arc-disjoint spanning trees rooted at a given vertex of the form $(\alpha, \alpha, \ldots, \alpha, \alpha)$ with $\alpha \in \{0, \ldots, d-1\}$. Next we generalize this construction.

PROPOSITION 5.2. For all $\alpha \in \{0, \ldots, d-1\}$, *there exist $d-1$ arc-disjoint spanning trees of optimal depth $D+1$ rooted at vertex $(\alpha, \alpha, \ldots, \alpha, \alpha)$ of $B(d, D)$.*

Proof. Let $\alpha \in \{0, \ldots, d-1\}$. With each $\nu \in \{0, \ldots, d-1\}$, $\nu \neq \alpha$ we associate a tree T_ν consisting of the arc from $(\alpha, \ldots, \alpha)$ to $(\alpha, \ldots, \alpha, \nu)$ and the shortest-paths spanning tree rooted at $(\alpha, \ldots, \alpha, \nu)$ from which the vertex $(\alpha, \ldots, \alpha)$ which is a leaf, is deleted. Let $\nu \neq \mu$, and consider any arc $e_\nu = [\alpha, \ldots, \alpha, \nu, W_1]$ from T_ν and any arc $e_\mu = [\alpha, \ldots, \alpha, \mu, W_2]$ from T_μ where $0 \leq |W_i| \leq D$, $i = 1, 2$. The letters α, ν, μ are all distinct, thus $e_\nu \neq e_\mu$ and T_ν and T_μ are arc disjoint.

It is easy to verify that the depth of these trees is $D+1$. This is optimal since there is only one path of length $\leq D$ from $(\alpha, \ldots, \alpha)$ to any vertex $(x_1, \ldots, x_D)$, $x_1 \neq \alpha$. □

THEOREM 5.3. *For any vertex u of $B(d, D)$, there exist $d-1$ arc-disjoint spanning trees rooted at u of depth at most $D+2k+1$, where k is the length of the second longest subsequence of identical letters in u, that is, of depth at most $D + 2\lfloor \frac{D}{2} \rfloor + 1$.*

Proof. Let $u = (u_1, \ldots, u_D)$ be a given node of $B(d, D)$. We construct $d-1$ arc-disjoint spanning trees rooted at u in two steps. First we will use the shortest-paths trees rooted at vertices at distance $k+1$ from u where k depends on the form of u. Then, if an arc is common to two trees, we will remove this arc from one tree and replace it with another arc that does not appear elsewhere. During all these replacements we will keep the tree structure. We give an example of two arc-disjoint spanning trees of $B(3, 2)$ rooted at $(1, 0)$.

1. *Definition of k and α:* Let us consider a longest subsequence of u consisting of identical letters, and let α be this letter. Let k be the length of the second longest subsequence of identical letters. For example, if $u = (0, 0, \ldots, 0, 0)$, then $\alpha = 0$ and $k = 0$; if $u = (0, 0, 0, 1, 1, 1)$, then $\alpha = 0$ or 1 and $k = 3$; if $u = (0, 0, 0, 1, 1, 2, 2)$, then $\alpha = 0$ and $k = 2$; if $u = (0, 1, 0, 1, 0, 1)$, then $\alpha = 0$ or 1 and $k = 1$. For $u = (1, 0)$, $k = 1$ and we choose $\alpha = 0$.

2. *Definition of T_ν, $\nu \neq \alpha$:* For all $\nu \in \{0, \ldots, d-1\}$, $\nu \neq \alpha$, we build a tree T_ν. We start with the shortest-path P_ν between $u = (u_1, \ldots, u_D)$ and the node $(u_{k+2}, \ldots, u_D, \nu, \ldots, \nu)$ where the letter ν is repeated $k+1$ times, and we add the shortest-paths tree rooted at

$(u_{k+2}, \ldots, u_D, \nu, \ldots, \nu)$. Then we delete from it all the vertices appearing in P_ν (except $(u_{k+2}, \ldots, u_D, \nu, \ldots, \nu)$) and the subtrees rooted at them, and we reattach these subtrees directly to P_ν. Thus we have obtained a spanning tree T_ν rooted at u. In $B(3, 2)$, from $u = (1, 0)$ and our choice of $\alpha = 0$, there are two paths P_1 and P_2:

$$P_1 = (1, 0) \longrightarrow (0, 1) \longrightarrow (1, 1),$$

(2)

$$P_2 = (1, 0) \longrightarrow (0, 2) \longrightarrow (2, 2),$$

and the two trees T_1 and T_2 are shown in Fig. 3 (T_1 is the black tree, and T_2 is the grey tree). They are not disjoint, since $(1, 0) \longrightarrow (0, 2)$ belongs to both trees.

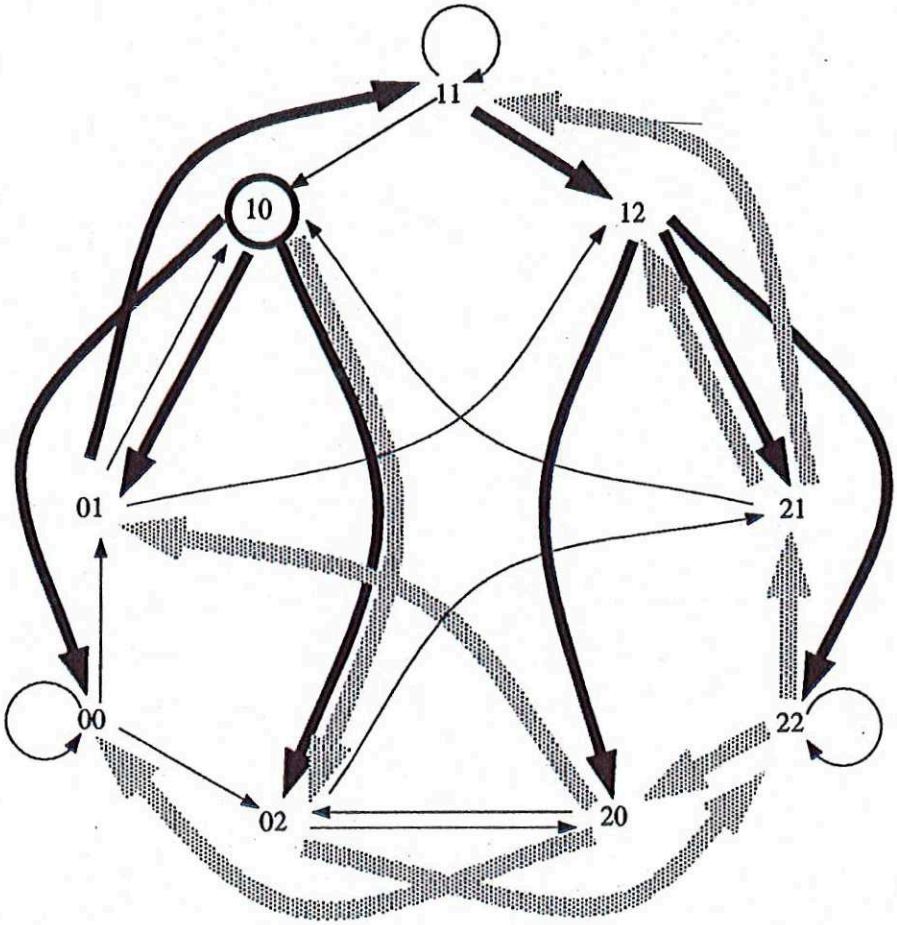

FIG. 3. *Construction of two arc-disjoint spanning trees rooted at* (10) *in* $B(3, 2)$.

3. *Definition of* T'_ν, $\nu \neq \alpha$: An arc of T_ν is of the form

(a) $$e_\nu = \left[u_i, \ldots, u_D, \overbrace{\nu, \ldots, \nu}^{\geq 1}{}^{\leq k+1} \right] \text{ if it belongs to } P_\nu,$$

(b) $$\text{otherwise } e_\nu = \left[u_i, \ldots, u_D, \overbrace{\nu, \ldots, \nu}^{k+1}, W \right] \text{ where } W \text{ is non-empty and } i \leq D,$$

(c) $$\text{or } e_\nu = \left[\overbrace{\nu, \ldots, \nu}^{\geq 1}{}^{\leq k+1}, W \right] \text{ where } W \text{ is non-empty.}$$

Let $\mu \neq \nu$ and $e_\nu = e_\mu$ where e_ν and e_μ are arcs of T_ν and T_μ, respectively. Since $\mu \neq \nu$, e_ν and e_μ cannot be both of type (a) or both of type (c). Suppose e_ν is of type (a) or (b), and e_μ of type (b). Then $e_\nu = [u_i, \ldots, u_D, \nu, \ldots, \nu, W]$ where W may be empty (and there may be less than $k+1$ consecutive letters ν), and

$$e_\mu = \left[u_j, \ldots, u_D, \overbrace{\mu, \ldots, \mu}^{k+1}, W' \right]$$

where W' is non-empty and $j \leq D$. If W is empty (type (a)), then, since $\mu \neq \nu$, there are $k+1$ consecutive letters μ in the subword $u_i, \ldots, u_D$, contradicting the definition of k as $\mu \neq \alpha$. If W is non-empty, then we can suppose without loss of generality that $j > i (i \neq j$ since $\mu \neq \nu)$. Thus, as before, there would be $k+1$ consecutive letters μ in the subword $u_i, \ldots, u_D$, a contradiction. Note that the same argument shows that e_μ cannot be of type (c) with $k+1$ letters μ at the beginning.

Thus exactly one arc is of type (c) with not more than k identical letters at the beginning. Suppose without loss of generality that it is $e_\mu = [\mu, \ldots, \mu, W]$ (W is non-empty). Moreover, this arc is also of type (a) or (b) in T_ν, that is, of the form $[u_i, \ldots, u_D, \nu, \ldots, \nu, W']$ (W' may be empty), $\nu \neq \mu$. We denote this kind of arc e_μ^ν. Such an arc satisfies

$$\begin{aligned} e_\mu^\nu &= [\mu, \ldots, \mu, W], W \text{ non-empty} \\ &= [u_i, \ldots, u_D, \nu, \ldots, \nu, W'], W' \text{ can be empty.} \end{aligned} \tag{3}$$

We will replace this arc in T_μ by the arc f_μ^ν which is obtained from e_μ^ν by replacing the first occurrence of μ by α.

Let us call T'_μ the new graph obtained from T_μ by doing all the possible replacements.

In the example in Fig. 3, the arc $[1, 0, 2]$ belongs to both T_1 and T_2. This arc, denoted e_1^2, is of type (c) since it can be written $[1, W]$ (an arc of T_1) and of type (a) since it can be written $[u_1, u_2, 2]$ (an arc of T_2). This arc is replaced in T_1 by the arc $f_1^2 = [\alpha, W] = [0, 0, 2]$ to build T'_1. The other tree is not modified: $T'_2 = T_2$. One can check on Fig. 3 that T'_1 and T'_2 are arc disjoint.

4. *The T'_μ, $\mu \neq \alpha$ are arc disjoint*: The arc f_μ^ν comes from the replacement of $e_\mu = e_\nu$ and so can be written

$$\begin{aligned} f_\mu^\nu &= [\alpha, \mu, \ldots, \mu, W], W \text{ non-empty} \\ &= [\alpha, u_{i+1}, \ldots, u_d, \nu, \ldots, \nu, W'], W' \text{ can be empty.} \end{aligned} \tag{4}$$

First, let us show that $f_\mu^\nu \neq f_{\mu'}^{\nu'}$ for all $\mu \neq \mu'$. Necessarily, $\nu \neq \nu'$, otherwise there would be two arcs entering to the same node in T_ν, namely $e_\mu^\nu = [\mu X]$ and $e_{\mu'}^\nu = [\mu' X]$ where $X = (u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W')$. We have

$$\begin{cases} f_\mu^\nu = [\alpha, u_{i+1}, \ldots\ldots, u_D, \nu, \ldots, \nu, W], \\ f_{\mu'}^{\nu'} = [\alpha, u_{j+1}, \ldots\ldots, u_D, \nu', \ldots, \nu', W'], \end{cases} \tag{5}$$

with at least one ν and one ν'. Without loss of generality, we can suppose that $j > i$ (therefore $i+1 \leq D$), W' is non-empty, otherwise W would also be empty and this would imply $\nu = \nu'$. Therefore $f_{\mu'}^{\nu'}$ contains a subsequence of $k+1$ letters ν'. Thus $\alpha, u_{i+1}, \ldots, u_D$ contains also a subsequence of $k+1$ letters ν' contradicting the definition of k (since $\alpha \neq \nu'$).

Finally, f_μ^ν cannot be an arc of T_γ, $\gamma \neq \mu$. Indeed, assume that $f_\mu^\nu = e_\gamma$:

$$f_\mu^\nu = [\alpha, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W] \tag{6}$$

and

$$e_\gamma = \begin{cases} [u_j, \ldots, u_D, \gamma, \ldots, \gamma] \text{ (type (a))} \\ \text{or} \\ [u_j, \ldots, u_D, \gamma, \ldots, \gamma, W'] \text{ (type (b))} \\ \text{or} \\ [\gamma, \ldots, \gamma, W''] \text{ (type (c))}. \end{cases} \tag{7}$$

The arc e_γ cannot be of type (c) because $\gamma \neq \alpha$. If $\gamma \neq \nu$, a similar argument shows that there is a subsequence of $k+1$ letters γ or ν in $u = (u_1, \ldots, u_D)$ contradicting the definition of k. But $\gamma = \nu$ implies that the node $(u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W)$ is reached in T_ν by the arc $[\alpha, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W]$. This is impossible since it is reached in T_ν by the arc $e_\nu = [u_i, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W] = e_\mu^\nu$, and $u_i \neq \alpha$ since $e_\mu^\nu = [\mu, \ldots]$.

5. *The T'_μ, $\mu \neq \alpha$ are connected*: It suffices to show that the path in T'_μ from the root to the tail of the new arc $[\alpha, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W']$ is identical to the corresponding path in T_μ. Suppose that some arc of this path has been modified. Then it must be of the form $[\mu, \ldots, \mu, \alpha, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W'']$ (because the path must reach the tail of $[\alpha, u_{i+1}, \ldots, u_D, \nu, \ldots, \nu, W']$), and it must also be of the form $[u_j, \ldots, u_D, \gamma, \ldots, \gamma, W''']$ for some $\gamma \neq \mu$.

If $\gamma \neq \nu$, then a similar argument as before shows that there is a subsequence of $k+1$ letters γ or ν in $u = (u_1, \ldots, u_D)$ contradicting the definition of k. So $\gamma = \nu$, but it again implies a contradiction with the definition of k by looking carefully at the positions of u_i in the two forms of the arc and noting that $\nu \neq \alpha$, $u_i \neq \alpha$, and $u_i \neq \nu$ (recall that in fact $u_i = \mu$).

6. *The depth of the T'_μ is at most $D + 2\lfloor \frac{D}{2} \rfloor + 1$*: The path from the root to any vertex in T_μ is of length at most $k + 1 + D$. In the replacement process, we might attach at most one (by step 5) subtree of depth at most $k - 1$. Indeed, the root of this subtree is the head of an arc e_μ^ν, which is (by step 3) of the type (c) but cannot contain more than k letters μ. Altogether the total depth is at most $k + 1 + D + 1 + k - 1 = D + 2k + 1$. We always have $k \leq \lfloor \frac{D}{2} \rfloor$. □

The proof of Theorem 5.3 is constructive and gives a method to construct arc-disjoint spanning trees of $B(d, D)$. With the notation of the proof, first find α and k. Next build the $d-1$ shortest-paths trees rooted at the $d-1$ vertices $(u_{k+2}, \ldots, u_D, \nu, \ldots, \nu)$, $\nu \neq \alpha$ (where the letter ν is repeated $k+1$ times). Finally, correct the trees following the described rule. Concerning the depth, note that $k = \lfloor \frac{D}{2} \rfloor$ is a worst case that occurs with a small probability. In general, k is much smaller than $\lfloor \frac{D}{2} \rfloor$, hence the depth of the arc-disjoint spanning trees is much smaller than $D + 2\lfloor \frac{D}{2} \rfloor + 1$. For instance, if the root is $u = (\alpha, \alpha, \ldots, \alpha, \alpha)$, our construction is the one given in Proposition 5.2 and we have a depth of $D + 1$.

COROLLARY 5.4. *Let r be any node of $B(d, D)$. There exists a protocol of broadcasting from r whose time is at most $(\sqrt{L\tau/(d-1)} + \sqrt{2DB})^2$.*

For a large message, this time is of the same order as the lower bound given by Proposition 3.1, i.e., $b_{B(d,D)}(r) \geq \max(D\beta, L\tau/(d-1)\tau)$.

6. Gossiping in networks. First we consider a given digraph $G = (V, E)$ of diameter D and minimum in-degree $d^-_{\min} = \min_{u \in V} d^-(u)$, and next we study the particular case of

the de Bruijn digraph. Gossiping is broadcasting from all the nodes. We assume that all the messages are of the same length L.

6.1. General lower bounds. Since during any gossiping all the nodes must perform a broadcast, the total start-up time is at least $\max_{r\in V} ecc(r)\beta = D\beta$. Let u be any node of V and S be a set of vertices not containing u. All the $|S|$ messages initiated by the vertices in S must reach u through $m^+(S, V - S)$ communication links, hence the total propagation time is at least

$$\max_{u\in V} \max_{S\neq\emptyset \mid u\notin S} \frac{|S|L}{m^+(S, V - S)}\tau. \tag{8}$$

For instance, choosing $S = V - \{u\}$ the total propagation time is at least

$$\max_{u\in V}((n - 1)L/d^-(u))\tau = ((n - 1)L/d^-_{\min})\tau.$$

PROPOSITION 6.1. *In a digraph G of minimum in-degree $d^-_{\min}$ and diameter D, the gossiping time is at least $g_G \geq \max(D\beta, ((n - 1)L/d^-_{\min})\tau)$.*

Note that we cannot obtain a lower bound by adding the two lower bounds (start-up and propagation time) [13], [31].

According to the above reasoning and particularly to (8), a good gossiping algorithm might proceed in D steps, ensuring that for any set of vertices S maximizing the ratio $|S|/m^+(S, V - S)$, the messages crossing from S to $V - S$ use the $m^+(S, V - S)$ links with a well-balanced load of the messages on the links. Johnsson and Ho [20] and MacKenzie and Seidel [25] show that this is possible for hypercubes and star graphs, respectively. We will show that this is also possible for the de Bruijn digraphs.

6.2. Gossiping in digraphs. We give below a simple greedy algorithm that appears to reach an optimal propagation time in the de Bruijn digraph. More details concerning the algorithm and its extensions can be found in [3], [4], [23]. We define the receiving phase as follows: receive while data arrive through any link and while all the links have not transmitted at least one message. With this convention, we describe following gossiping algorithm.

ALGORITHM (the algorithm is given for a processor u)
Step i ($1 \leq i \leq D$):

i. Form a message called *New* consisting of all the messages u has not already sent (at step 1, *New* will consist of the message of u itself). Send *New* to all the out-neighbors of u.
ii. Wait until you receive all the messages from your in-neighbors.

Note that to ensure that the algorithm works, u has to send a message at each phase i. If u has not received any new messages at phase $i - 1$, it can send an empty message or a special one.

The algorithm applied to the de Bruijn digraph is shown in Fig. 4. (The messages are numbered with the name of the processor in decimal arithmetic.)

The following lemma shows that the algorithm realizes the total exchange of the messages in D steps and will enable us to compute an upper bound of the gossiping time.

LEMMA 6.2. *At step i, each processor p forwards all the messages from all processors p' such that there exists a shortest path of length $i - 1$ from p' to p ($d(p', p) = i - 1$).*

Proof. Let us call P_i the property "*at step i, each processor p forwards all the messages from all processors p' such that $d(p', p) = i - 1$.*" We proceed by induction. P_1 is true. Assume that P_j is true for all $j < i$. During step $i - 1$, each processor q sends to all its

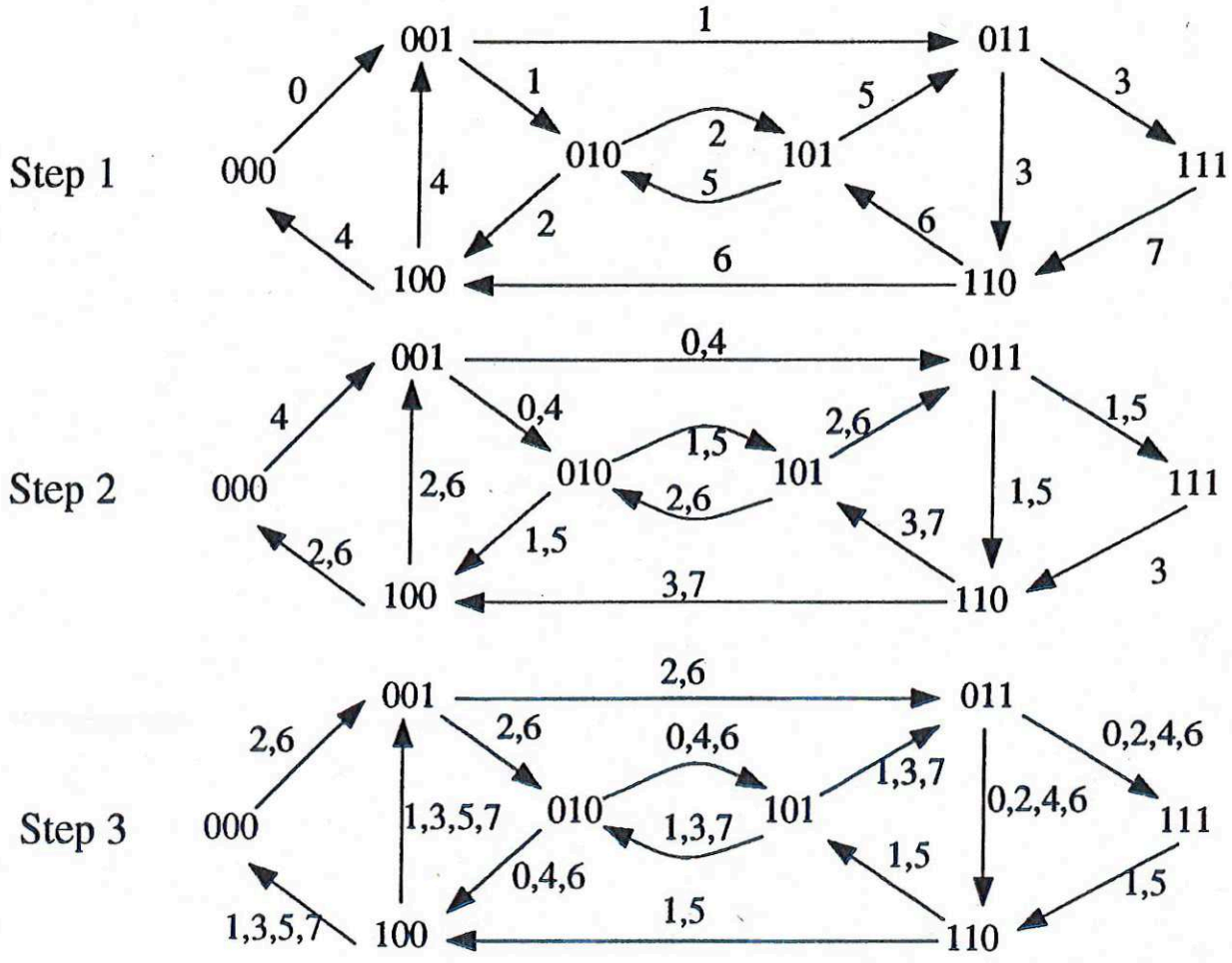

FIG. 4. *Gossiping in* $B(2, 3)$.

neighbors all the messages from processors p' such that $d(p', q) = i - 2$ by the induction hypothesis. Thus all the messages from processors p' such that $d(p', p) \leq i - 1$ have been received by p at the end of step $i - 1$. During step i, p forwards messages from processors p' such that $d(p', p) = i - 1$ since it had already forwarded those from processors p' such that $d(p', p) < i - 1$ by hypothesis. Thus P_i is true. □

Let $d^-_{\max}$ be the maximum in-degree of the considered network. From a processor p, there are at most $(d^-_{\max})^i$ processors p' such that $d(p', p) = i$; thus the maximum time of step i of the greedy algorithm is less than $\beta + (d^-_{\max})^{i-1} L\tau$ assuming that all messages are of same length L. Hence:

$$T_{Greedy} \leq \sum_{i=1}^{D} \left(\beta + (d^-_{\max})^{i-1} L\tau\right) \tag{9}$$

$$= D\beta + \frac{(d^-_{\max})^D - 1}{d^-_{\max} - 1} L\tau.$$

THEOREM 6.3. *In any digraph G of diameter D and maximum in-degree $d^-_{\max}$, there exists a protocol of gossiping whose running time is at most*

$$D\beta + \frac{(d^-_{\max})^D - 1}{d^-_{\max} - 1} L\tau.$$

For short messages, the greedy algorithm runs in $\Theta(D\beta)$, which is optimal. But depending on the considered network, the upper bound of the greedy algorithm cost can be far from the

lower bound of the time to gossip. However, we will show in the next section that, for the de Bruijn digraph, this upper bound is of the same order as the lower bound for large messages.

6.3. Gossiping in de Bruijn digraphs. For a de Bruijn digraph $B(d, D)$, $d^-_{\min} = d - 1$, thus $g_G \geq (n - 1/d - 1)L\tau$. Moreover, $d^-_{\max} = d$ and $n = d^D$, thus the greedy algorithm applied to the de Bruijn digraph has a complexity $T_{Greedy} \leq D\beta + (n - 1/d - 1)L\tau$.

COROLLARY 6.4. *In the de Bruijn digraph $B(d, D)$, there exists a protocol of gossiping whose running time is at most $D\beta + (n - 1/d - 1)L\tau$.*

For large messages, this protocol is of the same order as the lower bound. Moreover, for any length of message, it is at most two times slower than an optimal algorithm since the lower bound given by Proposition 6.1 is max $(D\beta, ((n - 1)L/d - 1)\tau)$. Note that Fig. 4 shows that there may exist a dissymmetry on the load of the arcs during each step of the greedy algorithm. Moreover, note that there are small redundancies in the transmission of the data. Hence, it may be possible to decrease the global cost by a small amount.

7. Case of undirected graphs. Designers prefer to construct networks based on undirected graphs. Indeed, layouts of mono- and bidirectional links are of the same complexity. If there is a link between u and v, then a message can be sent directly from u to v and from v to u. Recall that if only one of these messages can be sent at any given time, it is a *half duplex* mode of communication. Otherwise it is a *full duplex*. Here we are interested in the full duplex communication mode and so it is better to think in terms of symmetric digraphs. We can therefore apply the results above. We will now examine only what is happening for the undirected de Bruijn graph $UB(d, D)$.

A node $u = (u_1, \ldots, u_D) \in UB(d, D)$ is linked with $(u_2, \ldots, u_D, \alpha)$ and $(\alpha, u_1, \ldots, u_{D-1})$. A very important point is that we do not remove any of the double bidirectional links between nodes. For instance, (010) is linked twice in $UB(2, 3)$ with (101) by two bidirectional links. The maximum degree of $UB(d, D)$ is $2d$, but its minimum degree is $2d - 2$. We will use $UB^*(d, D)$ to denote the symmetric digraph obtained from $UB(d, D)$.

7.1. Broadcasting. Consider the set T_v of $d - 1$ arc-disjoint trees as constructed in §5. They use the arcs of $UB^*(d, D)$ in only one direction. Moreover, we can consider another family S_v of $d - 1$ arc-disjoint spanning trees by using right shifts instead of left shifts. Therefore, we have constructed a family of $2d - 2$ arc-disjoint spanning trees of $UB^*(d, D)$ of depth h at most $2D + 1$. Thus using Theorem 3.3, we have:

COROLLARY 7.1. *In any symmetric de Bruijn graph $UB^*(d, D)$ there exists a protocol whose broadcasting time is at most $(\sqrt{L\tau/2d - 2} + \sqrt{2D\tau})^2$.*

For large messages, this time is of the same order as the lower bound given by Proposition 3.1, i.e., $b_{UB^*(d,D)}(r) \geq \max(D\beta, (L/2d - 2)\tau)$.

7.2. Gossiping. We easily deduce from the greedy algorithm of §6 a greedy algorithm for $UB^*(d, D)$. Each message is divided into two parts. One part is diffused using the original gossiping (with left shifts), whereas the second part is diffused performing a similar gossiping with right shifts. The complexity is then:

$$
\begin{aligned}
T_{Greedy} &\leq \sum_{i=1}^{D} \left(\beta + d^{i-1}\frac{L}{2}\tau\right) \\
&= D\beta + \frac{d^D - 1}{d - 1}\,\frac{L}{2}\tau \qquad (10) \\
&= D\beta + \frac{n - 1}{2d - 2}L\tau.
\end{aligned}
$$

COROLLARY 7.2. *In the de Bruijn graph $UB^*(d, D)$, there exists protocol of gossiping whose running time is at most $D\beta + \frac{n-1}{2d-2}L\tau$.*

As for $B(d, D)$, for large messages this protocol is of the same order as the lower bound, and for any length of messages it is at most two times slower than an optimal algorithm since the lower bound given by Proposition 6.1 is $g_{UB^*(d,D)} \geq \max(D\beta, \frac{(n-1)L}{2d-2}\tau)$.

8. Conclusion. The de Bruijn digraphs (or undirected graphs) are a good family for performing broadcasting or gossiping. Our broadcasting and gossiping algorithms have propagation times that reach the optimal order for large messages. To do this, we have constructed a family of arc-disjoint spanning trees. These trees have a small depth and are easy to construct when the root is a vertex of the form $(\alpha, \ldots, \alpha)$. This confirms the interest of using these vertices as gates with the outside world. It will be interesting to improve the depth of the spanning trees constructed in general. It remains to study the same problem for other families such as Kautz digraphs or generalized de Bruijn or Kautz graphs. Finally, the general problem of bounding the maximum depth of arc-disjoint spanning trees in general graphs is interesting in itself and, as far as we know, algorithms for constructing arc-disjoint spanning trees of minimum depth have not been yet proposed (this problem has been proved to be NP-complete by Noga Alon; see sketch of his proof in [1]).

Acknowledgments. We are grateful to Joseph Peters and Dominique Sotteau for many helpful discussions and remarks.

REFERENCES

[1] J.-C. BERMOND AND P. FRAIGNIAUD, *Broadcasting and NP-completeness*, Graph Theory Notes of New York, XXII (1992), pp. 8–14.

[2] J.-C. BERMOND, N. HOMOBONO, AND C. PEYRAT, *Large fault-tolerant interconnection networks*, Graphs and Combinatorics, 5 (1989), pp. 107–123.

[3] J.-C. BERMOND AND J.-C. KONIG, *General and efficient decentralised protocols* 2, in International Workshop on Parallel and Distributed Algorithms, Bonas (1988), Elsevier-North Holland, Amsterdam, 1989, pp. 199–210.

[4] J.- C. BERMOND, J.-C. KONIG, AND M. RAYNAL, *General and efficient decentralised consensus protocols*, Second International Workshop on Distributed Algorithms, Amsterdam, 1987, Lecture Notes in Comput. Sci. 312, New York, Springer-Verlag, 1988, pp. 41–56.

[5] J.-C. BERMOND AND C. PEYRAT, *Broadcasting in de Bruijn networks*, in Proceedings of the 19th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congr. Numer., 66 (1988), pp. 267–282.

[6] ———, *de Bruijn and Kautz networks: a competitor for the hypercube?*, in Hypercube and Distributed Computers, F. ANDRE AND J. VERJUS, eds., Elsevier-North Holland, Amsterdam, 1989, pp. 279–294.

[7] R. CYPHER, *Theoretical aspects of VLSI pin limitations*, TR 89-02-01, Dept. of Comput. Sci., University of Washington, Seattle, 1989.

[8] W. DALLY AND C. SEITZ, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Trans. Comput., c-36 (1987), pp. 547–553.

[9] N. DE BRUIJN, *A combinatorial problem*, Koninklijke Nederlandse Academie van Wetenschappen Proc., A49 (1946), pp. 758–764.

[10] J. EDMONDS, *Edge-disjoint branchings, combinatorial algorithms*, in Combinatorial Algorithms. R. Rustin, ed., Algorithmics Press, New York, 1972, pp. 91–96.

[11] P. FRAIGNIAUD, *Performance analysis of broadcasting in hypercubes with restricted communication capabilities*, J. Parallel Dist. Comput., 16 (1992), pp. 15–26.

[12] P. FRAIGNIAUD AND E. LAZARD, *Methods and problems of communication in usual networks*, Discrete Appl. Math. (special issue on broadcasting), to appear.

[13] P. FRAIGNIAUD, S. MIGUET, AND Y. ROBERT, *Scattering on a ring of processors*, Parallel Comput., 13 (1990), pp. 377–383.

[14] D. GUSFIELD, *Connectivity and edge-disjoint spanning trees*, Inform. Proc. Lett., 16 (1983), pp. 87–89.

[15] S. T. HEDETNIEMI, S. HEDETNIEMI, AND A. LIESTMAN, *A survey of gossiping and broadcasting in communication networks*, Networks, 18 (1986), pp. 319–349.

[16] M. HEYDEMANN, J. OPATRNY, AND D. SOTTEAU, *Broadcasting and spanning trees in de Bruijn and Kautz networks*, Discrete Appl. Math. (to appear).
[17] W. HILLIS, *The Connection Machine*, MIT Press, Cambridge, MA, 1985.
[18] C. HO, *Full bandwidth communications on folded hypercubes*, in Proceedings International Conference on Parallel Processing, 1990.
[19] M. IMASE, T. SONEOKA, AND K. OKADA, *Fault-tolerant processor interconnection networks*, Systems Comput. Japan, 17 (1986), pp. 21–30.
[20] S. JOHNSSON AND C.-T. HO, *Optimum broadcasting and personalized communication in hypercubes*, IEEE Trans. Comput., 38 (1989), pp. 1249–1268.
[21] P. KERMANI AND L. KLEINROCK, *Virtual cut-through: a new computer communication switching technique*, Computers Networks, 3 (1979), pp. 267–286.
[22] S. KUNDU, *Bounds on the number of disjoint spanning trees*, J. Combin. Theory, 17 (1974), pp. 199–203.
[23] T. LAKSHMAN AND W. WEI, *Efficient decentralized consensus protocol using specially structured communication graphs*, Technical Report Bellcore (submitted to IEEE Trans. Comput.), 1990.
[24] L. LOVÁSZ, *On two minimax theorems in graph theory*, J. Combin. Theory, Ser B, 21 (1976), pp. 96–103.
[25] D. MACKENZIE AND S. SEIDEL, *Broadcasting on three multiprocessor interconnection topologies*, CS-TR-89-01, Michigan Technical University, Houghton, 1989.
[26] P. MICHALLON, D. TRYSTRAM, AND G. VILLARD, *Optimal broadcasting algorithms on torus*, Technical report RR872-I-0192, LMC, INPG, Grenoble, 1992.
[27] C. ST. J. A. NASH-WILLIAMS, *Edge-disjoint spanning trees of finite graphs*, J. London Math. Sec., 36 (1961), pp. 445–450.
[28] Y. SAAD AND M. SCHULTZ, *Data communication in parallel architectures*, Parallel Comput., 11 (1989), pp. 131–150.
[29] M. SAMATHAM AND D. PRADHAN, *The de Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI*, IEEE Trans. Comput., 38 (1989), pp. 567–581.
[30] S. SEIDEL, *Circuit-switched vs. store and forward solutions to symmetric communication problems*, Proceedings of the 4th Conference on Hypercube Concurrent Computers and Application, 1989.
[31] Q. STOUT AND B. WAGAR, *Intensive hypercube communication, prearranged communication in link-bound machines*, J. Parallel Dist. Comput., 10 (1990), pp. 167–181.