HAL
archives-ouvertes.fr

Optimal Deadlock-free Path-based Multicast Algorithms in Meshes

Vincent Bouchitté, Johanne Cohen, Eric Fleury

▶ **To cite this version:**

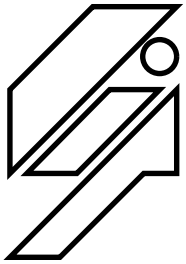
Vincent Bouchitté, Johanne Cohen, Eric Fleury. Optimal Deadlock-free Path-based Multicast Algorithms in Meshes. [Research Report] LIP RR-1997-31, Laboratoire de l'informatique du parallélisme. 1997, 2+13p. hal-02102049

HAL Id: hal-02102049
https://hal-lara.archives-ouvertes.fr/hal-02102049

Submitted on 17 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Laboratoire de l'Informatique du Parallélisme

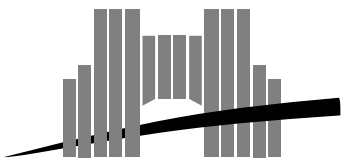
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

Optimal Deadlock-free Path-based Multicast Algorithms in Meshes

Vincent BOUCHITTÉ, Johanne COHEN and Eric FLEURY

October 1997

Research Report N^{O} 97-31



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) (0)4.72.72.80.00 Télécopieur : (+33) (0)4.72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

Optimal Deadlock-free Path-based Multicast Algorithms in Meshes

Vincent BOUCHITTÉ, Johanne COHEN and Eric FLEURY

October 1997

Abstract

Multicasting is an information dissemination problem which consists, for a node of a distributed memory parallel computer, of sending the same message to an arbitrary subset of nodes. The two major criteria to be considered in multicast communication are *traffic* (number of channels used) and *latency* (time required). In this paper, we proposed new polynomial algorithms giving optimal solutions in terms of traffic or latency for a mesh network using *wormhole* routing and *path-based* facility. All the algorithms are shown to be deadlock-free. Moreover, we generalize our algorithms to arbitrary Hamiltonian graphs.

Keywords: Multicasting, Wormhole Routing, Deadlock Free, Multiprocessors, Parallel Computers.

Résumé

Une *diffusion partielle* est une opération de communication sur une machine parallèle à mémoire distribuée dans laquelle un processeur veut envoyer un même message à un sous groupe de processeurs. Les deux critères dévaluation couramment employés sont le *traffic* (nombre de canaux utilisés) et la *latence* (temps requis). Dans ce rapport, nous proposons de nouveaux algorithmes polynomiaux calculant une solution optimale soit en terme de traffic, soit en terme de latence. Nous présentons ces algorithmes dans le cas d'une grille avec comme hypothèse que le mode de routage mis en œuvre est *wormhole* et utilise une facilité de routage nommée *path-based*. Tous les algorithmes sont sans interblocage. Nous généralisons nos algorithmes à la classe des graphes hamiltoniens.

Mots-clés: diffusion partielle, routage wormhole, machines parallèles, interblocage.

Optimal Deadlock-free Path-based Multicast Algorithms in Meshes

Vincent BOUCHITTÉ, Johanne COHEN and Eric FLEURY*

LIP - CNRS
École Normale Supérieure de Lyon
69364 Lyon Cedex 07, France

{vbouchit, jcohen, fleury}@lip.ens-lyon.fr
Tel: (33) 4 72 72 80 00 ; Fax: (33) 4 72 72 80 80

October 1997

Abstract

Multicasting is an information dissemination problem which consists, for a node of a distributed memory parallel computer, of sending the same message to an arbitrary subset of nodes. The two major criteria to be considered in multicast communication are *traffic* (number of channels used) and *latency* (time required). In this paper, we proposed new polynomial algorithms giving optimal solutions in terms of traffic or latency for a mesh network using *wormhole* routing and *path-based* facility. All the algorithms are shown to be deadlock-free. Moreover, we generalize our algorithms to arbitrary Hamiltonian graphs.
Key words: Multicasting, Wormhole Routing, Deadlock Free, Multiprocessors, Parallel Computers.

1 Introduction

Massively parallel computers (MPCs) are characterized by the distribution of memory among a set of processing nodes. Because they do not physically share memory, nodes in MPCs must communicate by passing messages through a communication network. Communications among the processors of a distributed memory parallel computer are often the main causes of performance degradations. Indeed, the "elementary communication time" is still larger than the "elementary computation time". Therefore it is of a main interest to focus on the development of efficient communication strategies for distributed memory multiprocessor. Many such commercial systems use cut-through switching techniques, called *wormhole routing* [2, 15]. In wormhole routing, a message consists of a sequence of *flits*. The header flit(s) of the message governs the route and other flits follow in a pipeline fashion through routers in the network.

Some communication operations are *point-to-point*, or *unicast*, in that they involve only a single source and a single destination. Other operations are *collective*, in that they involve more than two nodes. Our research addresses the design of collective communication operations. In this paper, we are interested in *multicast*, which is, possibly one of the most fundamental collective operations, in which data is delivered from one node to a designated subset of the nodes. One can remark that both unicast and *broadcast*, in which the destination set contains all the nodes in the networks, are special cases of multicast. In practice, parallel algorithms rarely allocate all nodes of a MPC, thus multicast operation is required, even in case where a process of an application needs to broadcast a message to all other processes of the application. Multicast appears in many parallel numerical algorithms and it is useful to support barrier synchronization, and cache coherency.

*The last author was supported by the LAVOISIER's program of the French Foreign Office. Eric FLEURY is a corresponding author. Mailing and e-mail addresses are given above.

Collective communication operations may be implemented either in software or hardware. In the first approach (called *unicast-based* collective communications [14]), collective algorithms are implemented atop send and receive primitives. We will focus on the second approach; routers are enhanced with generic hardware features which can be used to support collective operations. To reduce software overhead and improve collective algorithm performance, two techniques have been proposed: *message replication* and *intermediate reception*.

In message replication, the router is able to replicate message, flit-by-flit, and forward it onto several outgoing channels. This can be compared with an hardware extension of a multicast or broadcast tree. The main problem with message replication is that when any branch of the tree reaches a busy channel, the whole message must stop and occupying all channels used by the tree. Moreover, the dependencies that may occurre between branches may lead to deadlock [12].

We will therefore focus on the second technique. In intermediate reception, the router is able to copy the flits of an incoming message in the memory of the local processor, while simultaneously forwarding them to another outgoing link. A message (also called *multi-destination worm* [16]) can be sent by a source node and be routed as a "single worm" through several destination nodes, delivering a copy of the message at each of the intermediate destinations as it passes through. Such communication method is called *path-based* [11, 12].

Typically, a multi-destination worm contains in its header the list of destination node addresses it has to reach [1]. The order of these addresses corresponds to the order in which the destination nodes will be visited. As soon as a message header arrives at the first intermediate destination node of the list, the router associated with that node deletes its own address from the head of the list, and forwards the remainder of the message toward the next destination node.

A very important issue in wormhole-routed networks is to insure the absence of deadlock [9]. The routing algorithms used for multi-destination worms should provide enough degree of liberty in order to allow many intermediate destinations to be reached by a single worm. However, these routing algorithms must still prevent deadlock. One solution is to impose a total order on the use of channels. Lin and Ni [12] proposed to base routing decisions on a order imposed by a *Hamiltonian path* in the network. Using this method, a family of path-based multicast routing algorithms have been developed for mesh and hypercube networks [7, 11, 12]. This method can be used for any network containing a Hamiltonian path. Tseng and Panda [18] extended the path-based approach to networks that either do not contain Hamiltonian path or in which such a path may be unavailable, due to faults.

The two major criteria to be considered in multicast communication are the *traffic*, that is the number of channels used to deliver the source message to all its destinations, and the *latency* defined as the elapsed time from when the source sends out its first copy of the message until the last destination has received its copy of the message. Depending on the underlying communication paradigm and the routing method, multicast communication problems have been formulated as different theoretical graph problems. *Multicast Path Problem*: Find the shortest path starting from the source node and visiting all destination nodes; *Steiner Tree Problem*: Find a Steiner tree with a minimal total length; *Multicast Tree Problem*: Find a tree such that the distance from the source node to every destination is minimum. All the above problems have been shown to be NP-complete. The Steiner tree and Multicast path problems were shown to be NP-complete for regular topologies such as 2-D meshes and hypercubes [10, 12]. Also the Multicast Tree problem was shown to be NP-complete for hypercubes. Under the path-based communication model, the goal is to find a *multicast-star* ($\mathcal{MS}$), *i.e.*, a collection of several multicast paths. To minimize the traffic (first criterion) the goal is to minimize the total number of channels used by the $\mathcal{MS}$, and, for the latency (second criterion) the goal is to minimize the maximum length of the several paths of the $\mathcal{MS}$. It as been proved in [12] that finding an optimal multicast star according to the first criterion is a NP-complete problem for mesh and hypercube graphs.

In this paper, we show that, when basing the routing decision on an order imposed by a Hamiltonian path, as in [11] and [12], there exists polynomial time path-based multicast routing algorithms minimizing the number of channels used (criterion 1), or minimizing the maximum path lengths (criterion 2).

The remainder of this paper is organized as follows. Section 2 defines the system model for which our algorithms are designed. It also briefly gives necessary background information on wormhole routing. Section 3 reviews our algorithm minimizing the number of channels, and section 4 reviews our algorithm minimizing the maximum of the path lengths. Section 5 addresses generalization of both algorithms to

another class of graphs. Finally, we present our conclusion in section 6 where we discuss possible future areas of investigation.

2 Model and definitions

2.1 System model

In this paper, we consider massively parallel computers composed of nodes interconnected together by a fixed topology. The common element of nodes in recent multicomputers is a router, which determines the route of messages arriving, leaving and passing through the node. Figure 1 (taken from [15]) represents the architecture of a generic node. A router is connected to the local memory by pairs of *internal channels (input internal channel, output internal channel)*. In this paper, we suppose that routers are *all-ports*, that is: every pair of external channels have a corresponding pair of internal channels. The node can send to and receive from all its ports simultaneously. External channels are used to connect the router to neighboring routers, defining the topology of the network. As usual, we model the topology of the interconnection network by a graph $G = (V, E)$ in which each node in the set V corresponds to a processor node and each edge in the set E corresponds to a communication channel. Our discussion is mainly restricted to the *two-dimensional mesh* topology. This graph is formally defined as follow: a $m \times n$ mesh is a graph $G = (V, E)$ where $V = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid 0 \leq x < n \; and \; 0 \leq y < m\}$ and $E = \{((x_i, y_i), (x_j, y_j)) \mid |x_i - x_j| + |y_i - y_j| = 1\}$.

The predominant switching technique used in the last generation of MPC is the *wormhole routing*. We refer to [15] for a detailed survey on wormhole techniques. Roughly speaking, a message is divided into a number of *flits* for transmission. The header flit(s) of the message governs the route, and the remaining flits follow in a pipeline fashion. For wormhole routing, the network latency for sending a message of length L along a path of length d is:

$$\alpha + d\delta + (L - 1)\tau \tag{1}$$

where α is the startup time, δ is the time for a flit to be forwarded from a router to a neighboring one (commutation time plus the transfer time of one flit) and τ is the inverse of the bandwidth. Equation 1 does not take into account channel contentions that may appear on the route of the message when two messages want to use a same link in the same direction.

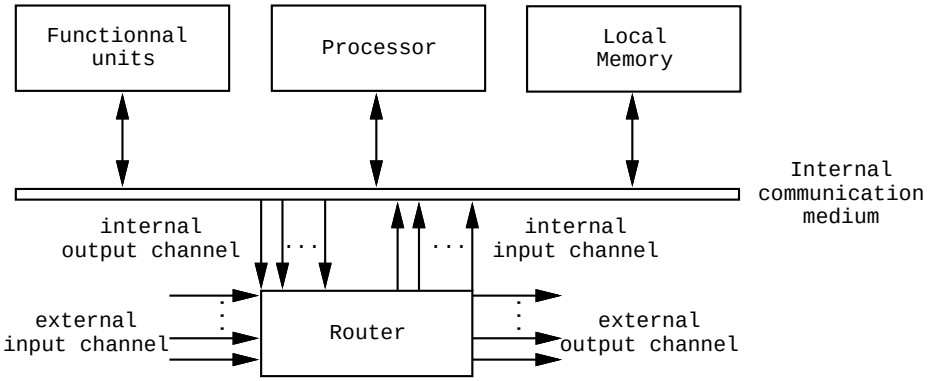


Figure 1: A generic node architecture.

We suppose that routers supports intermediate reception capability: the router is able to copy the flits of an incoming message to the memory of the local processor, while simultaneously forwarding them to another outgoing link. This feature allows the performance of hardware-supported multicast, called path-based routing [12]. The source forms a message, which is going to be routed through several destination nodes, delivering a copy of the message at each of the intermediate destinations as it passes through. Path-based multicasting can be done by building a list of destination addresses (according to the order in which they are to be visited) and setting the header to be this list of destinations, instead of being only one destination. Encoding the destination addresses in the header can be performed in several ways [1, 17].

2.2 Multicast and deadlocks

A *multicast-set* on a network $G = (V, E)$ is a tuple (u_0, K) where $u_0 \in V$ and $K = \{u_1, \ldots, u_k\}, u_i \in V, i = 1, \ldots, k$. The node u_0 is called the *source* node, and the k nodes of K are the *destination* nodes. Given a source node u_0, a *multicast* from u_0 is defined by a multicast-set (u_0, K), and corresponds to the information dissemination problem in which the node u_0 has to send the same message to all the nodes in K. All the previous works done by Lin et al. [11, 12] and Duato [4, 5, 6] use the concept of *split-and-sort function* whose aim is to prepare the multicast. Given a multicast-set (u, K), the split-and-sort function partitions the set of destinations into several subsets $K_1, K_2, \ldots, K_r$. The multicast algorithm consists then of sending one copy of the message to each subset. To do that, the split-and-sort function will build an appropriate header for each destination subsets by sorting the destinations according to the order in which they are to be visited. Each different way of splitting the destination set and sorting the destination subsets defines a different multicast algorithm.

Wormhole routing is very susceptible to deadlock since messages are allowed to hold many resources while requesting others. Hence, designing deadlock-free routing algorithms is an important issue. Multicast algorithms strongly depend on the routing function used to route messages. On a mesh, the most popular routing function is the XY-routing and consists of first routing in the X-dimension and then in the Y-dimension. When using unicast communication, this routing function is deadlock free [3] (the only type of channel dependencies created by this routing function is X to Y). However, using path-based hardware facility introduces new dependencies (more especially Y to X), and therefore the XY-routing is not deadlock-free anymore.

To overcome this problem, Lin and Ni [12] proposed to base routing decisions on a order imposed by an *Hamiltonian path* in the network. Each node u of a $m \times n$ mesh is assigned a label $\mathcal{L}(u)$ according to an Hamiltonian path: the labeling function $\mathcal{L}$ is the snake representation. More formally, the labeling function $\mathcal{L}$ for nodes of a $m \times n$ mesh is defined by:

$$\text{if } u = (x, y) \text{ then } \mathcal{L}(u) \quad = \quad \begin{cases} y * n + x & \text{if } y \text{ is even} \\ y * n + n - x - 1 & \text{if } y \text{ is odd} \end{cases}$$

where x and y are respectively the column and the row coordinates of the nodes. This labeling constructs an Hamiltonian path whose nodes are labeled consecutively from 0 to $mn - 1$. Figure 2 shows the Hamiltonian label assignment for a 3×4 mesh.

The route of a message is defined by a routing function $\mathrm{R} : V \times V \rightarrow V$ based on this Hamiltonian labeling. If u is the node currently holding the message and v is the destination node, then $\mathrm{R}(u, v) = w$ such that w is a neighboring node of u and:

$$\mathcal{L}(w) \quad = \quad \begin{cases} \max\{\mathcal{L}(z) \,|\, \mathcal{L}(z) \leq \mathcal{L}(v),\ z \text{ is a neighboring node of } u\} & \text{if } \mathcal{L}(u) < \mathcal{L}(v) \\ \min\{\mathcal{L}(z) \,|\, \mathcal{L}(z) \geq \mathcal{L}(v),\ z \text{ is a neighboring node of } u\} & \text{if } \mathcal{L}(u) > \mathcal{L}(v) \end{cases}$$

A message will visit its destination nodes sequentially according to R. In order to be deadlock-free, destinations must be visited in an monotonic order according to their label, allowing to maintain a monotonic order on channels. This implies that there cannot exist cyclic dependencies among the channels [12]. Given a multicast set (u_0, K), the split-and-sort function will first split K into two subset K_{inf} and K_{sup} where $K_{inf} = \{v \in K \,|\, \mathcal{L}(v) < \mathcal{L}(u_0)\}$ and $K_{sup} = \{v \in K \,|\, \mathcal{L}(v) > \mathcal{L}(u_0)\}$. The split-and-sort function then defines several subsets inside each K_{inf} and K_{sup} [8, 11, 12]. We suppose here, without loss of generality, that the label of destinations belonging to K are all greater than u_0 because, once the source has been chosen, the way the algorithm will treat K_{inf} or K_{sup} is symmetric.

2.3 Statement of the problem

The goal of a multicast algorithm is to find a collection of multicast-paths called *multicast-star*. Multicast-star may be defined as follows:

Definition 1 Let $G(V, E)$ be a graph, let $\mathcal{L}$ be a labeling of the nodes of this graph and R be a routing function requiring that destinations of messages are visited in a monotonic order according to the labeling

$\mathcal{L}$Let (u_0, K) be a multicast set where the degree of u_0 is Δ, a *multicast-star ($\mathcal{MS}$)* is a collection of multicast-paths $\{P_i\}$ described by the set of destination nodes $\mathcal{D}_i$ that they should reach, $1 \leq i \leq \Delta$, such that:

1. No destination node belongs to more than one multicast-path, *i.e.*, $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ if $i \neq j$;
2. The multicast-paths cover all the destinations, *i.e.*, $\bigcup_{1 \leq i \leq \Delta} \mathcal{D}_i = K$;
3. Destinations inside each multicast-path are visited in a monotonic order according to their labeling, *i.e.*, $\mathcal{L}(u_{i_j}) < \mathcal{L}(u_{i_{j+1}})$ if $u_{i_1} > u_0$ or $\mathcal{L}(u_{i_j}) > \mathcal{L}(u_{i_{j+1}})$ if $u_{i_1} < u_0$ where $u_{i_j} \in \mathcal{D}_i$ for $1 \leq i < \Delta$ and $1 \leq j < |\mathcal{D}_i|$;
4. The path P_i uses the output channel i of the source node u_0, $1 \leq i \leq \Delta$.

According to our system model, we can derive the communication time of a $\mathcal{MS}$ using equation 1. The communication time will be bounded by:

$$\alpha + (L-1)\tau + \delta \max_{1 \leq i \leq \Delta} \sum_{j=0}^{|\mathcal{D}_i|-1} d_{\mathrm{R}}(u_{i_j}, u_{i_{j+1}}), \text{ with } u_{i_0} = u_0,\ 1 \leq i \leq \Delta \tag{2}$$

where $d_{\mathrm{R}}(u, v)$ is the length of the path from u to v defined by the routing function R.

Efficient multicast algorithms involve several, often incompatible, requirements. The first one is to minimize the message *latency* defined as the elapsed time from when the source sends out its first copy of the message until the last destination has received its copy of the message. The second requirement is to minimize the amount of network *traffic*, that is minimize the number of channels involved in the multicast. Given a host graph $G(V, E)$, a labeling $\mathcal{L}$ of the nodes of this graph, a routing function R requiring that destinations of messages are visited in a monotonic order according to the labeling $\mathcal{L}$ and a multicast set (u_0, K) where the degree of u_0 is Δ, both minimization problems can be formulate as follows:

Problem 1 Find a *optimal channel multicast star ($\mathcal{OCMS}$)*. That is, a $\mathcal{MS}$ such that the the number of channels used is minimum:

$$\min_{\mathcal{MS}} \sum_{i=1}^{\Delta} \sum_{j=0}^{|\mathcal{D}_i|-1} d_{\mathrm{R}}(u_{i_j}, u_{i_{j+1}}) \text{ with } u_{i_0} = u_0,\ 1 \leq i \leq \Delta \tag{3}$$

Problem 2 Find a *optimal time multicast star ($\mathcal{OTMS}$)*. That is, a $\mathcal{MS}$ such that the total time given by equation 2 is minimum, which is equivalent to:

$$\min_{\mathcal{MS}} \max_{1 \leq i \leq \Delta} \sum_{j=0}^{|\mathcal{D}_i|-1} d_{\mathrm{R}}(u_{i_j}, u_{i_{j+1}}) \text{ with } u_{i_0} = u_0,\ 1 \leq i \leq \Delta \tag{4}$$

We are now going to describe our algorithms solving these two problems. Remember that in the following we suppose, without loss of generality, that the source label is smaller than the label of the destination nodes, *i.e.*, $\mathcal{L}(u_0) < \mathcal{L}(u_i)$, $1 \leq i \leq k = |K|$. Since we are considering a grid, the source node will construct at most two multicast-paths for the corresponding destination set. We note $\{\ell, \ell'\}$ these two output ports connecting the source u_0 to two neighboring nodes which have a greater label. If the source node has only one output port, the problem is obviously solved by the construction of only one multicast-path, obtained by sorting the destinations according to their labels.

3 An algorithm for finding an $\mathcal{OCMS}$ in a mesh

Before presenting our algorithm formally, we first give a description of how it works. In accordance with the hypothesis of definition 1, we are given a mesh $G = (V, E)$, a labeling $\mathcal{L}$, a routing function R and a multicast set (u_0, K). We want to construct an optimal channel multicast star using at most two paths. The algorithm is divided into three phases. The first phase builds a weighted bipartite graph, the second phase computes a minimum weighted perfect matching and the last phase extracts the multicast star from the matching.

We now introduce some definitions relevant to the notion of constraints, induced by both the labeling and the routing function, which will be needed in the formulation of our algorithm.

3.1 The constraint function c

Given a multicast set (u_0, K), we want to find multicast-paths satisfying the conditions of definition 1. T he source node u_0 plays an important role since this specific node has two available ports to "build" the two multicast-paths. For example, on Figure 2, the source node is allowed to send a message to node a via its "horizontal" output port ℓ (respectively to node b via its "vertical" output port ℓ') but it cannot send a message to node a using its "vertical" output port ℓ'. Moreover, we have to take into account the constraining conditions needed to guarantee a deadlock-free routing function, that is, destinations of a message must be visited in a monotonic order according to their label. Since we are considering only destinations which have a label greater than the source label, this means that from a node u_i, the message will be able to reach another node u_j only if $\mathcal{L}(u_i) < \mathcal{L}(u_j)$.

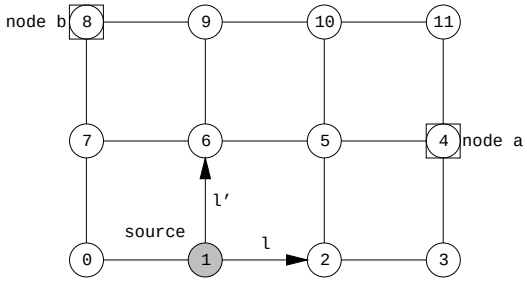


Figure 2: Label assignment for a 3×4 mesh.

To deal with this constraints, we introduce the constraint function $\mathrm{c} : K \times K \times E^* \to \mathbb{N}$ where $E^* = E \cup \{*\}$. The constraint function between two nodes u_i, u_j and relative to an output link ℓ is defined by:

$$\mathrm{c}(u_i, u_j, \ell) = \begin{cases} d_{\mathrm{R}}(u_i, u_j) & \text{if } \ell = * \text{ and } \mathcal{L}(u_i) < \mathcal{L}(u_j) \\ d_{\mathrm{R}}(u_i, u_j) & \text{if } \ell = (u_i, w), \mathcal{L}(u_i) < \mathcal{L}(u_j) \text{ and } \mathrm{R}(u_i, u_j) = w \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

In other terms, $\ell = *$, means that we do not specify the out-port. This will be in fact the case for each intermediate destination. On the other hand, $l = e$, $e \in E$, implies that we are considering the case of sending a message using the out-port e. To resume the behavior of this function: the constraint function $\mathrm{c}(v, w, l)$ yields ∞ if it is impossible to forward a message from v to w using the edge l as an out-port of the vertex v.

3.2 The bipartite graph

Given a multicast set (u_0, K), the key idea of the construction phase is to build a bipartite graph $\mathcal{B}_{(u_0,K)} = (Q, R, F, w)$ where Q and R are the two sets of vertices, F is the set of edges and w is a weight function. This bipartite graph models all the constraints imposed by the routing function R and/or induced by the labeling $\mathcal{L}$. More formally, let $\Omega(u_0)$ be the set of output links of the source node u_0 such that a multicast path using any of these links can reach at least one destination node in K. For a mesh, $|\Omega(u_0)|$ equals 1 or 2 depending

on the location of the source node u_0 in the mesh. We define the bipartite graph $\mathcal{B}_{(u_0,K)} = (Q, R, F, w)$ as follows:

- Each vertex u_i of K, $1 \leq i \leq k = |K|$, has one instance q_i in Q and one instance r_i in R;
- The source node u_0, will have $|\Omega(u_0)|$ instances $\{q^{\ell}_{0,\,\ell \in \Omega(u_0)}\}$ in Q. Theses $|\Omega(u_0)|$ instances represent the $|\Omega(u_0)|$ output links available from the source node u_0 to build the $|\Omega(u_0)|$ multicast-paths;
- In order to have the same number of vertices in Q and in R, we insert $|\Omega(u_0)|$ nodes $\{r_{k+i,\, i \in [1 \ldots |\Omega(u_0)|]}\}$ in R. Theses $|\Omega(u_0)|$ nodes are not involved in the multicast set. They will represent the ended extremities of the $|\Omega(u_0)|$ multicast-paths. The $|\Omega(u_0)|$ multicast-paths will end respectively at these $|\Omega(u_0)|$ nodes;
- There is a directed edge e from q^{ℓ}_0 to r_j in $\mathcal{B}_{(u_0,K)}$ if $c(u_0, u_j, \ell) \neq \infty$ and the weight of the edge e is set to $w(e) = c(u_0, u_j, \ell)$. That means that according to the routing function R, a message can go from u_0 to u_j using the output link ℓ, and the length of the path is given by $d_{\mathrm{R}}(u_0, u_j)$. Moreover, there is a directed edge e from q_i to r_j in $\mathcal{B}_{(u_0,K)}$ if $c(u_i, u_j, *) \neq \infty$ and the weight of the edge e is set to $w(e) = c(u_i, u_j, *)$. That means that, according the routing function R, a message can go from u_i to u_j;
- Every node $q \in Q$ is connected by a directed edge to every nodes in $\{r_{k+i,\, i \in [1 \ldots |\Omega(u_0)|]}\}$. The weight of all this edges, is set to be 0. That means that every node $u_i, 0 \leq i \leq k$, of the multicast set can be the last destination of the $|\Omega(u_0)|$ multicast-paths. Note that this includes the source node u_0.

If $|\Omega(u_0)| = 1$, then it implies that any $\mathcal{MS}$ of the multicast-set (u_0, K) is reduced to a single multicast-path and the solution of the problem becomes obvious. Now, we shall focus on the case where $|\Omega(u_0)| = 2$: to simplify the notation, we call q_0 and q'_0 the vertices in Q representing the node u_0. Now, we will present the basic properties of the bipartite graph $\mathcal{B}_{(u_0,K)}$.

Recall that a *matching* M of $\mathcal{B}_{(u_0,K)}$ is a subset of edges $M \subset F$ such that no two elements of M are adjacent. The matching is said to be *perfect* if every vertex in $Q \cup R$ is an end-point of some element of the matching M. The *weight* of the matching is given by $\sum_{e \in M} w(e)$.

Lemma 1 *$\mathcal{B}_{(u_0,K)}$ contains a perfect matching.*

Proof. Clearly, $\{q_0 r_1, q_1 r_2, \ldots, q_k r_{k+1}, q'_0 r_{k+2}\}$ is a perfect matching. □

Lemma 2 *Each $\mathcal{MS}$ of a given multicast set (u_0, K) is represented by a perfect matching of the bipartite graph $\mathcal{B}_{(u_0,K)}$, and conversely.*

Proof.

Given a $\mathcal{MS}$ of a multicast set (u_0, K), we build a matching M in the graph $\mathcal{B}_{(u_0,K)}$ as follows. Let P_1 and P_2 be the two multicast-paths of the $\mathcal{MS}$, and let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the set of destination nodes that they should reach respectively. According to Definition 1, the destination nodes of each path are visited in the monotonic order given by their label. We consider $\mathcal{D}_1 = \{u_{i_1}, \ldots, u_{i_s}\}$, and $\mathcal{D}_2 = \{u_{j_1}, \ldots, u_{j_t}\}$. We define a perfect matching of $\mathcal{B}_{(u_0,K)}$ by $M = \{(q_0, r_{i_1}), (q_{i_1}, r_{i_2}), \ldots, (q_{i_{s-1}}, r_{i_s}), (q_{i_s}, r_{k+1})\} \cup \{(q'_0, r_{j_1}), (q_{j_1}, r_{j_2}), \ldots, (q_{j_{t-1}}, r_{j_t}), (q_{j_t}, r_{k+2})\}$. We suppose here, without loss of generality, that the multicast-path P_1 uses the output link ℓ of the source node u_0 corresponding to the node q_0 in Q.

By construction of $\mathcal{B}_{(u_0,K)}$, all the edges of M exist. We now show that M is a matching. Clearly, there is only one edge of M incident to the vertices q_0, q'_0, r_{k+1} and r_{k+2}. Now consider a vertex q_i (resp. r_i), $1 \leq i \leq k$. The vertex q_i (resp. r_i) represents the vertex $u_i \in K$. Since u_i appears only once in $\mathcal{D}_1 \cup \mathcal{D}_2$, u_i has at most one successor (resp. predecessor) in $\mathcal{D}_1 \cup \mathcal{D}_2$, then q_i (resp. r_i) is incident to at most one edge of M. Thus M is a matching. Moreover, the number of edges of the matching M is $s + t + 2 = k + 2$ and since we know that $|Q| = |R| = k + 2$, it follows that M is a perfect matching.

Conversely, we consider a perfect matching M of the graph $\mathcal{B}_{(u_0,K)}$. Such a matching exits from Lemma 1. Let M be a matching represented by the set $\{(q_0, r_{i_1}), (q'_0, r_{i_2}), \ldots, (q_k, r_{i_{k+2}})\}$. From the matching M, we define a directed graph $H = (K \cup \{u_0\}, U)$ where the set U equals to $\{(u_0, u_{i_1}), (u_0, u_{i_2})\} \cup \{(u_i, u_j) \mid (q_i, r_j) \in M \text{ and } r_j \notin \{r_{k+1}, r_{k+2}\}\}$.

Let $d^+(x)$ (resp. $d^-(x)$) be the number of successors (predecessor) of the vertex x in the directed graph H. As M is a matching and thanks to the definition of the graph H, it is easily to check that:

- $d^+(u_0) = 2$ and $d^-(u_0) = 0$;
- $\forall i \in [1, \ldots, k]$, $d^-(u_i) = 1$;
- $d^+(u_k) = 0$;
- there exits a single integer j in $[1, \ldots, k]$, $d^+(u_j) = 0$ (corresponding to the edge (q_j, r_l) of the matching M where $l \in \{k+1, k+2\}$;
- $\forall i \in [1, \ldots, k[$ and $i \neq j$, $d^+(u_i) = 1$.

Thus, the graph that we obtaine consists of two disjoint paths but vertex u_0, and it covers all vertices of K. In fact, each path contains those vertices which are either in the destination nodes of a multicast-paths or in he other one but not both. It follows that we can deduce a multicast-star from a graph H. □

Theorem 1 *Each $\mathcal{OCMS}$ of a given multicast set (u_0, K) induces a minimum weighted perfect matching of the bipartite graph $\mathcal{B}_{(u_0,K)}$, and conversely.*

Proof. According to lemma 2, it is sufficient to notice that the weights of the $\mathcal{MS}$ and of the perfect matching associated with this $\mathcal{MS}$ are the same. □

3.3 The algorithm

Now, we describe our algorithm which is decomposed in three phases, namely the construction phase, the computation phase and the extraction phase. Initially, we execute the construction phase (see section 3.2). This phase constructs the bipartite graph $\mathcal{B}_{(u_0,K)}$ corresponding to the multicast-set (u_0, K) and to the routing function R. Once the construction phase is achieved, we execute the computation phase which computes a minimum-weight perfect matching M of the bipartite graph $\mathcal{B}_{(u_0,K)}$ using the method described in [13]. Finally, the extraction phase constructs the two sets of destination nodes $\mathcal{D}_1$ and $\mathcal{D}_2$ of the two multicast-paths P_1 and P_2 respectively. The correctness of Algorithm 1 is given by theorem 1.

Algorithm 1 $\mathcal{OCMS}(u_0, K)$

```
    ▷ Construction phase
1   Construct the valuated bipartite graph B_(u0,K) = (Q, R, F, w)
    ▷ Computation phase
2   Compute a minimum weighted perfect matching M of B_(u0,K)
    ▷ Extraction phase
3   let target be a function from F to R such that if an edge e of F equals to tuple (q, r), then target(e) = r
4   let D1 = ∅
5   let e = (q0, r_i1) ∈ M
6       while (target(e) ∉ {r_k+1 r_k+2}) do
7           let D1 = D1 ∪ {u_j} such that r_j = target(e)
8           let e = (q_j, r_t) such that e ∈ M
9       end while
10  let D2 = ∅
11  let e = (q'0, r_i1) ∈ M
12      while (target(e) ∉ {r_k+1 r_k+2}) do
13          let D2 = D2 ∪ {u_j} such that r_j = target(e)
14          let e = (q_j, r_t) such that e ∈ M
15      end while
```

It is easy to see that this algorithm can be implemented so as to run in $O(k^3)$-time. More precisely, for each phase we have:

- The construction of the bipartite graph in line 1 can be done in $O(k^2)$ operations;
- The computation of a minimum-weight perfect matching M of the bipartite graph $\mathcal{B}_{(u_0,K)}$ in line 2 can be achieved in $O(k^3)$ operations using the method described in [13];
- And the extraction phase (lines 3-10) requires no more than $O(k)$ operations.

4 An algorithm for finding an $\mathcal{OTMS}$ in a mesh

The purpose of this section is to describe a polynomial time algorithm computing an optimal time multicast star ($\mathcal{OTMS}$) for a given multicast set (u_0, K). This problem is equivalent to minimize the maximum length of all the multicast-paths of the $\mathcal{MS}$. Our algorithm is composed of three phases, namely the initialization phase, the computation phase and the extraction phase.

First, using the same argument as in section 3, we do not consider the case where the source node u_0 has only one output link (*i.e.*, $\Omega(u_0) = 1$). So, we will assume in the following that the source node u_0 has two output links (*i.e.*, $\Omega(u_0) = 2$).

The key idea is to consider all the possible choices to perform a multicast from the source node u_0 to the set of destination nodes $K = \{u_1, \ldots, u_k\}$. More formally, we are interested in all the $\mathcal{MS}$ of a multicast set (u_0, K_i), where $K_i = \{u_1, \ldots, u_i\}, 1 \leq i \leq k = |K|$. In order to do this, we construct a set of vertices Q as follows:

- The source node u_0, will have $|\Omega(u_0)| = 2$ instances $\{u^{\ell}_{0,\,\ell \in \Omega(u_0)}\}$ in Q. These 2 instances represent the 2 output links available from the source node u_0 to build the $|\Omega(u_0)|$ multicast-paths. To simplify notations, these two instances of u_0 are called q_0 and q_1;
- Each vertex u_i of K, $1 \leq i \leq k = |K|$, has one instance q_{i+1} in Q.

We are going to consider a $\mathcal{MS}$ of a multicast-set (u_0, K_i) as a tuple $(i+1, j, l_1, l_2)$ such that

1. $i + 1 > j$;
2. One of these two multicast-paths has the node q_{i+1} (corresponding to node u_i) as its extremity. And its length is equal to l_1;
3. The other multicast-path has the node q_j (corresponding to node u_{j-1} if $j > 0$, otherwise u_0) as its extremity, and its length is equal to l_2.

First, let us consider the particular case where the multicast-set is equal to $(u_0, \emptyset)$. There is only one $\mathcal{MS}$ corresponding to this multicast-set and it is defined by the tuple $(1, 0, 0, 0)$. In other words, the $\mathcal{MS}$ is composed of one multicast-path such that its extremity is q_1, corresponding to vertex u^{ℓ}_0. It means that this multicast-path will use output link ℓ. Note that all multicast-stars of a given multicast-set (u_0, K) may be represented by a tuple $(|K| + 1, j, l_1, l_2)$. Indeed, one of the two paths must end at node with the greatest label, and the other path may end at any other node, included the source node.

4.1 Description of the algorithm

Our algorithm is based on a dynamic programming approach. We split our algorithm into three parts:

1. In a first step, we initialize a 4-dimensional array T, such that all the elements are equal to $False$ except the element $(1, 0, 0, 0)$ which is equal to $True$. We will see in the following how to bound the indexes of this array. To simplify the algorithm, we define the function $c' : Q \times Q \to \mathbb{N}$ as follows:

$$c'(q_i, q_j) = \begin{cases} \mathrm{c}(q_i, q_j, \ell) & \text{if } q_i = u^{\ell}_0 \text{ where } \ell \in \Omega(u_0) \\ \mathrm{c}(q_i, q_j, *) & \text{if } q_i \notin \{u^{l}_0, u^{l'}_0\} \end{cases} \tag{6}$$

2. In a second step, the array T is filled by Algorithm 2 below:

Algorithm 2 Fill(T,u,K)

```
    ▷ To fill array T
1   for i1 = 2 to k do
2       for i2 = 2 to i1 do
3           for k1, k2 = 2 to n do
4               if T(i1,i2,k1,k2)= True then
5                   T(i1 + 1,i2,k1+c'(q_{i1}, q_{i1+1}),k2) ← True
6                   T(i1 + 1,i1,k2+c'(q_{i2}, q_{i1+1}),k1) ← True
```

7 **end if**

The way the array is filled insures that the following relation holds:

$$T(X) = True \Leftrightarrow \text{there exists a } \mathcal{MS} \text{ defined by the tuple } X.$$

3. Finally, among all the $\mathcal{MS}$ of the multicast set (u_0, K) (be reminded that the $\mathcal{MS}$ are represented by elements $(|K|+1, j, l_{|K|}, l_j)$ of T such that $T(|K|+1, j, l_{|K|}, l_j) = True$), we find the two paths which minimize the maximum length.

In the next section, we will prove the correctness of this algorithm.

4.2 Correctness of algorithm

We shall show how to bound the indexes of the 4-dimensional array T. First, we should ask whether or not there exists a $\mathcal{MS}$ of a multicast set such that the length of one of the two multicast-paths is greater than the number of vertices of the graph G. Clearly, the answer is "no" since the routing function R does not allow a multicast-path to pass through a same vertex more than one time. Indeed, by definition, the routing function imposes the restriction that a message must pass through nodes in a monotonic order according to their labels. Thus the maximum length of a multicast-path is bounded by $N-1$ where $N = mn$ is the number of nodes of the graph G. It follows that all $\mathcal{MS}$ can be defined by a 4-tuple in the space $[0, \ldots, k]^2 \times [1, \ldots, N-1]^2$. Secondly, note that algorithm 2 fills the entire array T in the following way: the algorithm will compute the element X of T before the element X' of T if the tuple X' is greater than X considering the lexicographic order. It follows that a path will visit destinations nodes in a monotonic order according to their labels. Finally, to conclude the proof of the correctness of our algorithm, we show the following theorem:

Theorem 2 *If there exists a $\mathcal{MS}$ of a given multicast-set (u_0, K_i) such that*

(i) one extremity of the multicast-path P_1 is the vertex u_i and its length is equal to k_1;

(ii) one extremity of the multicast-path P_2 is the vertex u_{j-1} (if $j > 0$ otherwise u_0) and its length equals to k_2;

(iii) $K_i = \{u_1, \ldots, u_i\}$ where $\forall j \in [1, \ldots, i], u_j \in K$.

then, the element $(i+1, j, k_1, k_2)$ of the array T is equal to True, and conversely.

Proof. We shall show that if there exists a $\mathcal{MS}$ defined by $X = (i+1, j, k_1, k_2)$, then $T(X) = True$. We prove this property by induction on parameter i.

As the basis for our induction, let us consider the case $i = 0$. We note that the statement is trivially true since after the initialization phase, only the element $(1, 0, 0, 0)$ of T is equal to $True$. So, this multicast star is composed of two multicast-paths that contain only the source node u_0. Then, their length are equal to zero, and they correspond to a $\mathcal{MS}$ for the multicast-set $(u_0, \emptyset)$. Thus, for $i = 0$, the theorem holds. We assume then, that it is true whatever the integer value $s < i$;

Now let us consider $s = i$. Assume that, on the contrary to this property, there exists a $\mathcal{MS}$ defined by a tuple X such that $T(X) = False$ where $X = (i+1, j, l_1, l_2)$. Let P_1 and P_2 be the two multicast-paths of the $\mathcal{MS}$ defined by $X = (i+1, j, l_{i+1}, l_j)$. Without loss of generality, we can consider that u_{i-1} is in P_1. There are two cases:

(i) u_i is in P_1: we can split the path P_1 into two sub-paths $P_1' \cup P_{u_{i-1} \to u_i}$ where P_1' is a sub-path of P_1 from the source node u_0 to u_{i-1} and $P_{u_{i-1} \to u_i}$ is a sub-path of P_1 from u_{i-1} to u_i. By definition, it implies that these two paths P_1', P_2 form a $\mathcal{MS}$ of multicast-set (u_0, K_{i-1}) defined by the tuple $X' = (i, j, d(P_1'), l_j)$ where $d(P_1')$ is the length of path P_1'. Now, by the induction hypothesis, since $i - 1 < i$, the property holds: $T(X') = True$. Using line (5), we set $T(i, j, d(P_1') + c'(u_{i-1}, u_i), l_j)$ to $True$. Hence by definition of the function c', we have $(i, j, d(P_1') + c'(u_{i-1}, u_i), l_j) = X$ and we can not have $T(X)$ to be equal to $False$. So it leads to a contradiction.

(ii) u_i is in P_2: we apply the same argument as the previous case.

Both cases provides a contradiction so that it must be the case that if there exists a $\mathcal{MS}$ defined by a tuple X then $T(X) = True$.

Conversely, using the same arguments as previously, it is easy to see by definition that if $T(X) = True$, then there exists a $\mathcal{MS}$ defined by X. □

Thanks to theorem 2, after filling the array T, all elements $X = (k+1, i, l_{k+1}, l_i)$ in T such that $T(X) = True$, represent all possibilities of $\mathcal{MS}$ corresponding to the multicast-set (u_0, K). It remains to be found the minimum of the maximum length of this two paths among all these elements of T. It follows that our algorithm computes an $\mathcal{OTMS}$.

It is easy to see that the whole algorithm may be implemented so as to run in $O(k^2N^2)$ where N is the number of vertices in the mesh. More precisely, for each phase we have:

- The initialization phase and the filling phase of the array T, can be done in $O(k^2N^2)$ operations;
- The extraction of the solution is equivalent to searching for the maximum value among $O(kN^2)$ elements, and so the complexity of this part is equal to $O(kN^2)$.

5 Generalization

In the two previous sections 3 and 4, given a mesh, a labeling $\mathcal{L}$, a routing function R and a multicast-set (u_0, K), we have described two algorithms computing an $\mathcal{OCMS}$ and an $\mathcal{OTMS}$ respectively. Unfortunately, the topology of the networks was fixed and the two algorithms were described for a specific topology, *i.e.*, a mesh. The purpose of this section is to apply ours algorithms to a more general class of graphs. More precisely, we are going to consider Hamiltonian graphs. By considering an Hamiltonian path in a graph $G = (V, E)$, we construct a labeling function $\mathcal{L}$ of the nodes according to this path. Thus, a routing function R having the same properties as described in section 2.1 may be used on any Hamiltonian graphs.[1] Note that it does not imply that this kind of routing function uses minimal path, which was the case for the mesh. Indeed, any other Hamiltonian labeling of the mesh would have lead to a non-minimal path routing function.

First, we can notice that the two algorithms described in sections 3 and 4 and the proofs of their correctness are based on the constraint function c that should be adapted. We assume that k is the cardinality of the set K and $\Omega(u)$ is the set of output links of the node u such that a multicast path passing through u and using any of these links can reach at least one other destination node in K. Let d be a distance function depending on the routing function R in G: $d_{\mathrm{R}}(u_i, u_j, \ell)$ is the minimum number of links used by function R to transmit a message from u_i to node u_j using the out-port ℓ of u_i. So, the constraint function between two nodes u_i and u_j and an output link $\ell \in \Omega(u_i)$ of the node u_i is defined by:

$$\mathrm{c}(u_i, u_j, \ell) = \begin{cases} \min_{\ell' \in \Omega(u_i)} d_{\mathrm{R}}(u_i, u_j, \ell') & \text{if } \ell = * \\ d_{\mathrm{R}}(u_i, u_j, \ell) & \text{if } \ell = (u_i, w), \text{ and } \mathrm{R}(u_i, u_j) = w \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

Now, we can describe the modifications needed for our algorithms such that they compute the $\mathcal{OTMS}$ or the $\mathcal{OCMS}$ of a multicast-set (u_0, K).

To compute the $\mathcal{OCMS}$, we will apply the same algorithm but we slightly modify the construction of the bipartite graph $\mathcal{B}_{(u_0,K)} = (Q, R, F, w)$. Now, $|\Omega(u_0)|$ nodes in Q and R (2 nodes in section 3) are going to represent the source node u_0 and each of them is associated with one output link of u_0. The $|\Omega(u_0)|$ nodes represented u_0 in Q (resp. in R) are the beginning (resp. the end) of one multicast-path. The time complexity of the resulting algorithm is $O((k + |\Omega(u_0)|)^3)$ operations.

To compute the $\mathcal{OTMS}$, we can use the same algorithm as in section 4. Instead of considering a $\mathcal{MS}$ as a 4-tuple, we are going to consider it as a $2|\Omega(u_0)|$-tuple: the multicast star $\mathcal{MS}$ is composed of $|\Omega(u_0)|$ paths. The time complexity of the resulting algorithm is $O((kN)^{|\Omega(u_0)|})$. Unfortunately, the complexity of the algorithm depends on the number of output links of the source node. However, the complexity of this algorithm remains polynomial if the degree of the source node is constant.

[1] It is easy to note that the routing function R having the same properties as described in section 2.1 may be used on a network if and only if this network contains an Hamiltonian path. To prove it, it is enough to consider the case of a broadcast from a node having the smallest label in the network to all other nodes.

6 Conclusion

We have considered wormhole routing in a mesh network with path-based facility. Under this assumptions, two new polynomial algorithms computing an $\mathcal{OTMS}$ and an $\mathcal{OCMS}$ for a given multicast-set (u_0, K) in an Hamiltonian graph G have been proposed. These algorithms were the first deadlock-free optimal polynomial algorithms that are able to optimize the communication traffic or minimize the time required to perform path-based multicasts. Moreover, both algorithms can be used to any networks that contain Hamiltonian path, included n-dimensional meshes. One of the disadvantages of this two path-based multicast algorithms is that some contentions may occurred in the network under certain conditions since the paths generated by the algorithms are not necessarily edge disjoint. If two messages want to use the same channel, one must be delayed and the total communication time will be increased.

Possible extension of this work is open to investigation. Perhaps improvements can be made to the algorithms presented here if it is possible to build an $\mathcal{OTMS}$ and an $\mathcal{OCMS}$ such that the multicast-paths are edge-disjoint.

Also, the algorithms presented in this paper and their generalization are designed explicitly for Hamiltonian graph. One open question is to know if the same algorithms may be used on other networks providing deadlock-free path-based wormhole routing function. This is an interesting issue and deserves further study.

References

[1] C.-M. Chiang and L. M. Ni, *Multi-address encoding for multicast*, in Parallel Computer Routing and Communication, First International Workshop, (PCRCW '94), K. Bolding and L. Snyder, eds., no. 853 in Lecture Notes in Computer Science, Springer-Verlag, May 1994, pp. 146–160.

[2] W. J. Dally and C. L. Seitz, *The torus routing chip*, Distributed Computing, 1 (1986), pp. 87–196.

[3] ——, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Transactions on Computers, C-36 (1987), pp. 547–553.

[4] J. Duato, *A new theory of deadlock-free adaptive routing in wormhole networks*, IEEE Transactions on Parallel and Distributed Systems, 4 (1993), pp. 1320–1331.

[5] ——, *On the design of deadlock-free adaptative multicast routing algorithms*, Parallel Processing Letters (special issue on Algorithmic and Structural Aspects of Interconnection Networks), 3 (1993), pp. 321–333.

[6] ——, *A theory of deadlock-free adaptative multicast routing in wormhole networks*, IEEE Transactions on Parallel and Distributed Systems, 6 (1995), pp. 976–987.

[7] E. Fleury and P. Fraigniaud, *Strategies for multicasting in meshes*, in 23rd International Conference on Parallel Processing (ICPP '94), Aug. 1994.

[8] ——, *Analysis of deadlock-free path-based wormhole multicasting in meshes in case of contentions*, in 6th Symposium on the Frontiers of Massively Parallel Computing (Frontiers '96), Annapolis, Maryland, October 1996, IEEE.

[9] ——, *A general theory for deadlock avoidance in wormhole-routed networks*, in 10th International Conference on Parallel and Distributed Computing Systems (PDCS-97), New Orleans, Louisiana, October 1997.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Computer science / mathematics, W. H. Freeman and Company, 1979.

[11] X. Lin, P. K. McKinley, and L. M. Ni, *Deadlock-free multicast wormhole routing in 2D-mesh multicomputers*, IEEE Transactions on Parallel and Distributed Systems, 5 (1994), pp. 793–804.

[12] X. Lin and L. M. Ni, *Deadlock-free multicast wormhole routing in multicomputer networks*, in 18th Annual International Symposium on Computer Architecture, Toronto, May 1991, ACM, pp. 116–125. (see also Technical Report MSU-CPS-ACS-29, 1990).

[13] L. Lovasz and M. D. Plummer, *Matching theory*, North-Holland, 1986.

[14] P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni, *Unicast-based multicast communication in wormhole-routed networks*, IEEE Transactions on Parallel and Distributed Systems, 5 (1994), pp. 1252–1265.

[15] L. M. Ni and P. K. McKinley, *A survey of wormhole routing techniques in direct networks*, IEEE Computers, 26 (1993), pp. 62–76.

[16] D. K. Panda, S. Singal, and P. Prabhakaran, *Multidestination message passing mechanism conforming to base wormhole routing scheme*, in Parallel Computer Routing and Communication (PCRCW '94), K. Bolding and L. Snyder, eds., no. 853 in Lecture Notes in Computer Science, Springer-Verlag, May 1994, pp. 131–145.

[17] C. B. Stunkel, R. Sivaram, and D. K. Panda, *Implementing multidestination worms in switch-based parallel systems: Architectural alternatives and their impact,*, in The 24th Annual IEEE/ACM International Symposium on Computer Architecture (ISCA '97), Denver, Colorado, June 1997.

[18] C. Tseng and D. K. Panda, *A trip-based multicasting model for wormhole-routed networks with virtual channels*, in 7th International Parallel Processing Symposium, Newport Beach, California, Apr. 1993, IEEE, pp. 276–283.