ELSEVIER

Information Processing Letters 73 (2000) 221–225

Information Processing Letters

www.elsevier.com/locate/ipl

Acyclic orientations do not lead to optimal deadlock-free packet routing algorithms ☆

Daniel Štefankovič [1]

Department of Computer Science, Comenius University, Bratislava, Slovakia

Received 21 August 1998
Communicated by S. Zaks

Abstract

In this paper we consider the problem of designing deadlock-free shortest-path routing algorithms. A design technique based on acyclic orientations has proven to be useful for many important topologies, e.g., meshes, tori, trees and hypercubes. It was not known whether this technique always leads to algorithms using an asymptotically optimal number of buffers. We show this is not the case by presenting a graph of size $\mathcal{N}$ which has a deadlock-free shortest-path routing algorithm using O(1) buffers, but every deadlock-free shortest-path routing algorithm based on acyclic orientations requires $\Omega(\log \mathcal{N} / \log\log \mathcal{N})$ buffers.

Keywords: Deadlock-free packet routing; Acyclic orientations; Interconnection networks

1. Introduction

Devising deadlock-free routing algorithms is an important problem in the network design. A wide range of different algorithms and design techniques has been proposed [1,4–11]. In this paper we concentrate on an important class of deadlock-free routing algorithms called *buffer-reservation* algorithms [2, 3]. Given source, destination and current buffer of a packet a buffer-reservation algorithm specifies a set of buffers to which the packet may move. The packet can move to any buffer from the specified set provided that it is empty.

It is known that for each network there exists a deadlock-free routing algorithm using only 2 buffers per node [8]. However this algorithm suffers from congestion, because it uses only a spanning tree of the network. It is reasonable to consider routing algorithms using only shortest paths. For such algorithms it was shown in [2] that there exists a network of size $\mathcal{N}$ which requires $\Omega(\log\log \mathcal{N})$ buffers per node. It is also known that for every network of diameter $\mathcal{D}$ there exists a shortest-path deadlock-free routing algorithm using $\mathcal{D}+1$ buffers per node [9].

A technique for constructing deadlock-free routing algorithms based on acyclic orientations has proven to be useful for many important topologies [9]. Moreover only few additional bits of data besides routing information are required by the algorithms designed using this technique. Thus a natural question is how good the technique is in general. We present a graph of size $\mathcal{N}$ which has deadlock-free shortest-path routing algorithm using O(1) buffers per node, but every deadlock-free shortest-path routing algorithm based

☆ This research has been partially supported by VEGA 1/4315/97.

[1] Email: stefanko@zeus.dcs.fmph.uniba.sk.

0020-0190/00/$ – see front matter
PII: S0020-0190(00)00022-3

on acyclic orientations requires $\Omega(\log\mathcal{N}/\log\log\mathcal{N})$ buffers per node. The proofs presented in this paper can be adapted to obtain the same result for the cube-connected-cycles graph.

2. Preliminaries

A communication network is modeled by an undirected graph $G = (V, E)$, where vertices in V represent nodes and edges in E represent bidirectional communication links. Every edge is considered to comprise two opositely oriented arcs.

Given source, destination and the buffer currently holding the packet a buffer-reservation algorithm specifies a set of buffers to which the packet may move. The packet can move to any of the specified buffers, provided it is empty. A *deadlock* is a situation in which a set of packets can never reach their destination, because specified buffers of each packet are occupied by other packets from the set. A buffer reservation algorithm is called *deadlock-free* if it does not allow a deadlock to occur. A routing algorithm is called *shortest path* if the packets are always routed along shortest paths.

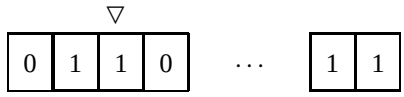
Given a graph G an *all-to-all path system* $\mathcal{P}$ is a collection of paths connecting every two vertices in G. An *orientation* DG of G is a directed graph obtained from G by replacing each undirected edge in G by an arc (i.e., $\{u, v\}$ is replaced by either (u, v) or (v, u)). An orientation is called *acyclic* if it doesn't contain a cycle. An *orientation cover* of a path system $\mathcal{P}$ is a sequence of orientations $DG_1, \ldots, DG_s$ such that every path $p \in \mathcal{P}$ can be written as a concatenation of s paths $p_1, \ldots, p_s$ where p_i is a path in DG_i for $i \in [s]$. An *acyclic orientation cover* is an orientation cover consisting of acyclic orientations.

Let $\mathcal{P}$ be a shortest-path all-to-all path system of a graph G. If an acyclic orientation cover for $\mathcal{P}$ of size s exists, then there exists a shortest-path deadlock-free routing algorithm using only s buffers per vertex [9]. A routing algorithm designed using this technique is said to be *based on acyclic orientations*. If there is no restriction on how a routing algorithm was designed then it is said to use *plain buffers*.

3. A gap between acyclic orientations and plain buffers

In this section we present a graph which has shortest-path deadlock-free routing algorithm using only O(1) plain buffers per vertex, but every shortest-path deadlock-free routing algorithm based on acyclic orientations requires $\Omega(\log\mathcal{N}/\log\log\mathcal{N})$ buffers per vertex.

Consider the following machine M_n:



It has a working tape with n cells and a head which can be positioned above any cell. Each cell contains one binary digit. In one step the head can change the content of the scanned cell (shuffle) or it can move to the left or to the right neighboring cell. The *state* of the machine M_n is the position of the head and the content of the tape.

Definition 1. Let G_n be the graph with vertices corresponding to states of the machine M_n and edges connecting two vertices iff the corresponding states are reachable in one step of M_n.

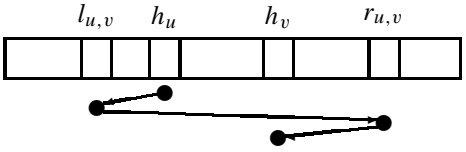
Remark 2. If the tape of the machine was wraparound the resulting graph would be the cube-connected-cycles graph. It is possible to show the same results for this graph with slightly more complicated proofs and worse constants.

3.1. Upper bound for plain buffers

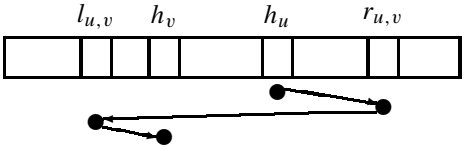
Theorem 3. *There exists a shortest-path deadlock-free routing algorithm of the graph G_n using only* 8 *buffers per vertex.*

Proof. Let u, v be any two vertices of the graph G_n. Moving along a shortest path from u to v corresponds to changing the state u to the state v using the minimal number of steps. Let h_u and h_v be positions of the head in u and v. Moreover let $l_{u,v}$ and $r_{u,v}$ be positions of the leftmost and the rightmost cell that have to be visited (every cell in which tapes of u and v differ as well as cells at positions h_u and h_v must be visited).

In every shortest path the head makes a movement in one of the following two forms:



CASE 1 ($h_u \leqslant h_v$)



CASE 2 ($h_u \geqslant h_v$)

Thus according to the head movement each path can be divided into three phases (the first and/or third phase can be empty) in which the head moves in one direction. Now we show that eight buffers per vertex suffice. Each vertex has buffers with names $1, 2, 3, 4, 1', 2', 3', 4'$. A packet moving along a path with the head movement—CASE 1—is initially put in the buffer 1 and then it moves between buffers according to these rules:

– left-arc (phase 1) any buffer $\rightarrow$ buffer 1
– right-arc (phase 2) any buffer $\rightarrow$ buffer 2
– left-arc (phase 3) any buffer $\rightarrow$ buffer 3
– shuffle-arc (all phases) buffer a $\rightarrow$ buffer a'

No shortest path contains two consecutive shuffle-edges and thus the rule for the shuffle-arc is correct (a packet that wants to move along a shuffle-arc cannot be in a buffer with a prime). A packet moving along a path with the head movement—CASE 2—is initially put in buffer 2 and moves between buffers according to these rules:

– right-arc (phase 1) any buffer $\rightarrow$ buffer 2
– left-arc (phase 2) any buffer $\rightarrow$ buffer 3
– right-arc (phase 3) any buffer $\rightarrow$ buffer 4
– shuffle-arc (all phases) buffer a $\rightarrow$ buffer a'

For the sake of contradiction suppose that there is a deadlock. Then there must be a sequence such that

$$(v_1, b_1) \rightarrow (v_2, b_2) \rightarrow \cdots \rightarrow (v_m, b_m) \rightarrow (v_1, b_1),$$

where $(u, a) \rightarrow (v, b)$ means that a packet from the buffer a in the vertex u waits for the buffer b in the vertex v. Observe that a packet never moves to a buffer with smaller number. Therefore all buffers $b_1, \ldots, b_m$ must have the same number—some $a \in [4]$. When a packet moves to a buffer a' it uses shuffle-arc and when it moves to a buffer a it uses a left-arc if a is odd and a right-arc if a is even. Therefore all buffers $b_1, \ldots, b_m$ must be a'. However a packet never moves from a' to a', a contradiction. □

3.2. Lower bound for acyclic orientations

The vertex set of an n-dimensional *hypercube* consists of all binary strings of length n. Two vertices are connected by an edge iff they differ in exactly one bit. For any two vertices u, v a movement along path from u to v corresponds to a sequence of changes of some bits. If the bits are changed in order from left to right then the path is called *monotone*. The lower bound for acyclic orientations depends on the following lemma:

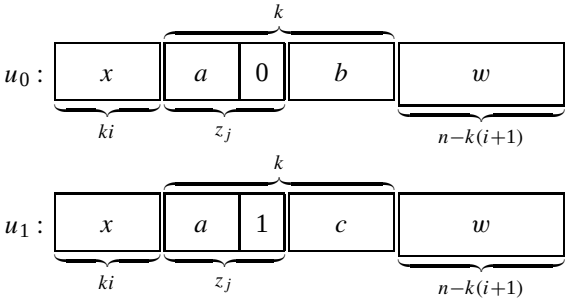
Lemma 4. *Let $\mathcal{P}$ be the all-to-all path system of an n-dimensional hypercube with monotone paths. Every orientation cover for $\mathcal{P}$ has size $\Omega(n/\log n)$.*

Proof. Let $DG_1, \ldots, DG_s$ be an orientation cover for $\mathcal{P}$. We show that the size of this cover must be large by simulating movement of packets in the network. Each vertex generates a packet for every other vertex. The packets then move along their paths according to the orientation DG_1, then according to DG_2 and so on and finally after moving according to DG_s every packet must be in its destination. We observe the state of the network (positions of packets) after each orientation.

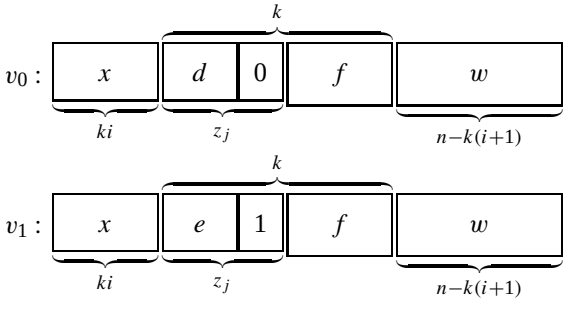
Let d be a number $0 \leqslant d \leqslant n$ and v be a vertex. By S_v^d we denote the set of all vertices that can differ from v only in the first d bits and by D_v^d the set of all vertices that can differ from v only in the last $n - d$ bits. Given a state of the network we call the vertex v *active* with respect to d if for each vertex $u \in D_v^d$ there exists a packet which has to move to u through v. Clearly if v has not yet received some packet $\langle p \rangle$ from some vertex $w \in S_v^d$ then v is active with respect to d, because packets from w to every vertex in D_v^d travel together with $\langle p \rangle$ until they reach v (the paths are monotone). A vertex which is not active with respect to d is called *passive* with respect to d. Clearly a vertex passive with respect to d must have already received all packets from all vertices in S_v^d.

Let $k = \log_2 n$. We will construct a sequence of hypercubes $Q_0, \ldots, Q_{n/k}$ such that after moving according to the ith orientation the ratio of passive vertices in Q_i with respect to ki is at most $i/2^k$. This would imply that, because if some vertices are active then some packets are not in their destination. For each i the hypercube Q_i will have dimension $n - ki$ and it will consist of vertices starting with some string from $\{0,1\}^{ki}$.

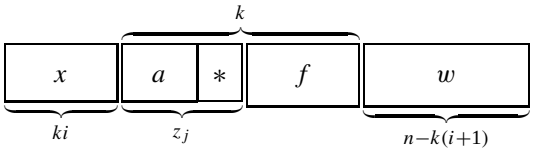
Initially all vertices are active and thus we can take Q_0 to be the whole hypercube. Now we show how to construct Q_{i+1} from Q_i. Let Q_i be an $(n - ki)$-dimensional sub-hypercube consisting of vertices starting with some string $x \in \{0,1\}^{ki}$ and suppose that the network is in state after moving according to the ith orientation. Let M denote the number of passive vertices (from Q_i) with respect to ki. For any $w \in \{0,1\}^{n-k(i+1)}$ let A_w denote the set of active vertices (from Q_i) with respect to ki such that they end with w. Now we move the packets along their paths according to the orientation DG_{i+1}. How many vertices from Q_i ending with w can be passive with respect to $k(i+1)$ after the movement? Let N_w denote the set of these vertices. Let $l \in [k]$ be such that $2^l \geqslant |N_w| > 2^{l-1}$ (the case $|N_w| \leqslant 1$ is treated separately). By induction it is easy to show that there exist distinct $z_1, \ldots, z_l \in [k]$ such that for each z_j, $j \in [l]$, there exist two vertices in N_w of the form:



where a, b, c are some binary strings. If there were two vertices in A_w of the form:



where d, e, f are some binary strings, then either a packet from v_0 to u_1 or a packet from v_1 to u_0 cannot move between source and destination in the orientation DG_{i+1}, because these two packets need to cross the edge between vertices:



in the opposite direction. This is a contradiction to our assumption that u_0 and u_1 will be passive after the movement and thus there cannot be such v_0, v_1 pair in A_w. Therefore for each $j \in [l]$ the $(ki + z_j)$th bit of each vertex from A_w is determined by bits $(ki + z_j + 1), \ldots, k(i+1)$. This implies that $|A_w| \leqslant 2^{k-l}$. For any $l \in [k]$:

$$|N_w| - \left(2^k - |A_w|\right) \leqslant 2^l - 2^k + 2^{k-l} \leqslant 1.$$

The inequality holds also in case $|N_w| \leqslant 1$. Summing the inequalities for each $w \in \{0,1\}^{n-k(i+1)}$ we obtain:

$$N - M \leqslant 2^{n-k(i+1)},$$

where N is the total number of vertices from Q_i passive with respect to $k(i+1)$ after the movement. Clearly there exists a string $q \in \{0,1\}^k$ such that at most $(M + 2^{n-k(i+1)})/2^k$ vertices starting with xq are passive with respect to $k(i+1)$ after the movement. Let Q_{i+1} be the hypercube consisting of vertices starting with xq. The ratio of passive vertices increased from

$$R_i = \frac{M}{2^{n-ki}} \quad \text{to}$$

$$R_{i+1} = \frac{(M + 2^{n-k(i+1)})/2^k}{2^{n-k(i+1)}} = R_i + \frac{1}{2^k}.$$

Therefore if in Q_i the ratio was at most $i/2^k$ then in Q_{i+1} the ratio will be at most $(i+1)/2^k$. □

Theorem 5. *Any shortest-path deadlock-free packet routing algorithm of the graph G_n based on acyclic orientations requires $\Omega(n/\log n)$ buffers.*

Proof. Let S be the set of all vertices from G_n having the head on the leftmost bit and D be the set of all vertices having the head on the rightmost bit. For every $u \in S$ and $v \in D$ there is a unique shortest path from

u to v in G_n. It corresponds to changing the bits in which u and v differ while moving the head to the right. Let $\mathcal{P}$ be the set of the shortest paths from every $u \in S$ to every $v \in D$. Let $C = DG_1, \ldots, DG_s$ be an acyclic orientation cover used to implement shortest-path deadlock-free packet routing algorithm on G_n. Clearly C is also an acyclic orientation cover for $\mathcal{P}$.

By merging the vertices having the same tape content we obtain the n-dimensional hypercube and the path system $\mathcal{P}$ becomes all-to-all path system of the n-dimensional hypercube with monotone paths. The acyclic orientation cover C becomes orientation cover of the new path system. Using Lemma 4 we obtain that the size of C must be $\Omega(n/\log n)$. □

A consequence of Theorems 3 and 5 is:

Theorem 6. *Graph G_n of size $\mathcal{N} = n2^n$ has a deadlock-free shortest-path routing algorithm using* $O(1)$ *buffers per node, but each deadlock-free shortest-path routing algorithm based on acyclic orientations requires $\Omega(\log\mathcal{N}/\log\log\mathcal{N})$ buffers per node.*

4. Conclusion

There are many results on buffer-reservation algorithms for specific topologies, but for general graphs very little is known. It would be interesting to have better upper and lower bounds on the number of buffers required for shortest-path deadlock-free routing on general graphs. Concerning acyclic orientations it would be interesting to know how large the gap between routing algorithms based on acyclic orientations and routing algorithms using only plain buffers can be.

Acknowledgement

Thanks are due to Ivona Bezáková, Rastislav Kráľovič, Branislav Rovan, Peter Ružička and Richard Tan for helpful discussions and for carefully reading the draft of this paper.

References

[1] B. Awerbuch, S. Kutten, D. Peleg, Efficient deadlock-free routing, in: Proc. 10th Annual ACM Symposium on Principles of Distributed Computing, 1991, pp. 177–188.

[2] R. Cypher, Minimal, deadlock-free routing in hypercubic and arbitrary networks, in: Proc. 7th IEEE Symposium on Parallel and Distributed Processing, 1995.

[3] R. Cypher, L. Gravano, Requirements for deadlock-free, adaptive packet routing, in: Proc. 14th Annual ACM Symposium on Principles of Distributed Computing, 1992, pp. 25–33.

[4] W.J. Dally, C. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, IEEE Trans. Comput. 36 (5) (1987) 547–553.

[5] J. Duato, Deadlock-free adaptive routing algorithms for multicomputers: Evaluation of a new algorithm, in: Proc. 3rd IEEE Symposium on Parallel and Distributed Processing, 1991.

[6] I.S. Gopal, Prevention of store-and-forward deadlock in computer networks, IEEE Trans. Commun. 33 (12) (1985) 1258–1264.

[7] K.D. Günther, Prevention of deadlocks in packet-switched data transport systems, IEEE Trans. Commun. 29 (4) (1981) 512–524.

[8] P.M. Merlin, P.J. Schweitzer, Deadlock avoidance in store-and-forward networks, IEEE Trans. Commun. 28 (3) (1980) 345–354.

[9] G. Tel, Deadlock-free packet routing, in: Introduction to Distributed Algorithms, Cambridge University Press, Cambridge, UK, 1994, Chapter 5.

[10] S. Toueg, K. Steiglitz, Some complexity results in the design of deadlock-free packet switching networks, SIAM J. Comput. 10 (4) (1981) 702–712.

[11] S. Toueg, J.D. Ullman, Deadlock-free packet switching networks, SIAM J. Comput. 10 (3) (1981) 594–611.