ELSEVIER

Information Processing Letters 71 (1999) 235–239

Information Processing Letters

www.elsevier.com/locate/ipl

A note on deflection worm routing on meshes

Antonios Symvonis [1]

Basser Department of Computer Science, University of Sydney, Sydney, N.S.W. 2006, Australia

Received 1 April 1997; received in revised form 1 February 1999
Communicated by R.G. Dromey

Abstract

We prove the existence of asymptotically optimal routing schedules for deflection worm routing of permutations on $n \times n$ meshes. We achieve this by deriving an off-line algorithm which routes permutations in $O(kn)$ steps where k is the number of flits of a worm. The best to date off-line algorithms for deflection worm routing of permutations were due to Newman and Schuster (1995) and Sibeyn and Kaufmann (on-line) (1994) which completed the routing in $O(k^{1.5}n)$ routing steps.

Keywords: Deflection routing; Mesh connected computer; Packet routing; Permutation routing; Worm routing; Parallel processing

1. Introduction

Message routing has been abstracted in several ways. In *packet routing* it is assumed that a message can be transmitted between two adjacent processors in a single step as a *packet*. If packets can be stored in intermediate nodes during the trip from their origin to their destination, the routing model is referred to as *store-and-forward*. In a different model known as *deflection* (or *hot-potato*) routing, packets continuously move between processors from the time they are injected into the network until the time they are consumed at their destination.

The advantage of deflection routing over the store-and-forward model is obvious. No queuing area is required at the processors. However, the fact that packets always move implies that at any step each processor must transmit the packets it received during the previous step (unless they were destined for it). As a result, several packets might be routed away from their destination. This makes the analysis extremely difficult.

The assumption that a whole message can be transmitted in a single step between adjacent processors is not a very realistic one, especially when the messages are long. It is more natural to assume that the amount of information that can be transmitted between processors in a single step is a hardware dependent variable (the *width* of the communication channel). This leads to the modeling of a message as a *worm*; a sequence of *k flits*, each of size equal to the width of the communication channel, in which only the first flit knows the destination address. During the routing of a worm, all routing decisions are made by the processors which hold the head of the worm. The rest of the flits (the *body* of the worm) simply follow the path of the head. When the worms are allowed to be queued at intermediate processors waiting for the release of a communication link, we say that routing is performed according to the *store-and-forward worm routing* model. When

[1] Email: symvonis@cs.su.oz.au.

0020-0190/99/$ – see front matter
PII: S0020-0190(99)00105-2

queuing is not allowed, the *deflection worm routing* model is used. See [8] for a good review of message routing in parallel machines.

In this research note, we concentrate on deflection worm routing of permutations on $n \times n$ meshes. Deflection worm routing on meshes was first examined by Bar-Noy et al. [1]. They studied permutation routing and presented $O(k^{2.5}n2^{O(\sqrt{\log n \log\log n})})$-step and $O(kn^{1.5})$-step deterministic and $O(kn)$-step randomized algorithms.

Newman and Schuster [3] described a method to obtain deflection worm routing algorithms based on store-and-forward packet routing algorithms. Their method was general enough to work for any routing patterns, not only permutations. By employing the sorting algorithm of Schnorr and Shamir [4] they obtained an $O(k^{2.5}n)$-step deflection worm routing algorithm for routing permutations. They also presented an $O(k^{1.5}n)$-step off-line algorithm. Newman and Schuster also observed that better results for routing permutations could be obtained if fast algorithms for one-to-many routing [2,5] or many-to-many routing [5] were available. Sibyen and Kaufmann [5] used such algorithms to derive an $O(k^{1.5}n)$-step deflection worm routing algorithm for permutations.

This research note makes a theoretical contribution to the literature of off-line deflection worm routing algorithms. We present an $O(kn)$-step off-line algorithm for routing permutations on $n \times n$ meshes. The existence of such an algorithm was implied by the $O(kn)$-step randomized algorithm of Bar-Noy et al. [1] through standard but not constructive arguments. The best off-line algorithm known till now was the $O(k^{1.5}n)$-step algorithm of Newman and Schuster [3]. Sibyen and Kaufmann [5] managed to achieve the same number of steps by an on-line algorithm. Note that an $O(kn)$-step solution to a permutation problem is asymptotically optimal as a standard bisection argument reveals an $\Omega(kn)$-step lower bound. Finding an $O(kn)$-step off-line algorithm was posed as an open problem in [3].

2. Preliminaries

A *finite directed graph* $G = (V, E)$ is a structure which consists of a finite set of vertices V and a finite set of edges $E = \{e_1, e_2, \ldots, e_{|E|}\}$. Each edge is incident to the elements of an ordered pair of vertices (u, v) where u is the start-vertex of the edge and v is its end-vertex. We will use the notation $E(G)$ and $V(G)$ to denote the edge set and the vertex set of graph G, respectively. A *directed path* is a sequence of edges $e_1, e_2, \ldots$ such that the end-vertex of e_{i-1} is the start-vertex of e_i. The set $\mathit{Neighbors}(v, G)$ is defined to be the set of vertices in G which can be reached from v by crossing just one edge. Formally,

$$\mathit{Neighbors}(v, G) = \big\{w \mid (v, w) \in E(G)\big\}.$$

Let M_n denote an $n \times n$ two-dimensional mesh. M_n has n^2 vertices, each represented by an ordered pair of integers $(\mathit{row}, \mathit{column})$ where $0 \leqslant \mathit{row}, \mathit{column} \leqslant n - 1$. Each vertex (i, j), $0 \leqslant i, j \leqslant n - 1$, has outgoing edges to its four neighbors $(i - 1, j)$, $(i + 1, j)$, $(i, j - 1)$, $(i, j + 1)$, provided that they exist.

A *deflection packet* (*worm*) *routing problem* R is defined by a tuple (G, P) where $G = (V, E)$ is the directed graph representing the network in which the routing will take place (vertices in V represent processors and edges in E represent unidirectional communication links). The elements in set P represent the m packets (worms) to be routed. Formally,

$$\begin{aligned} P = \big\{ & p_1, p_2, \ldots, p_m \mid \\ & p_i = (\mathit{orig}_i, \mathit{dest}_i), \\ & \mathit{orig}_i, \mathit{dest}_i \in V(G),\ 1 \leqslant i \leqslant m \big\}. \end{aligned}$$

Even though there is no restriction on the number of packets (worms) which originate from (or, are destined for) a certain processor, in this research note we focus on permutations, that is, exactly one packet (worm) originates from each node and exactly one packet (worm) is destined for each node.

Consider any routing problem $R = (G, P)$ where $G = (V, E)$ is the directed graph which represents the interconnection network in which the routing takes place and P is the set of worms to be routed. Our goal is to achieve routing time near the lower bound for the problem. Assume an upper bound of T routing steps for the problem under consideration (for now consider a worst case trivial upper bound).

We construct a *multistage directed graph* $G' = (V', E')$ as follows:

$$V' = \big\{(v, t) \mid v \in V \text{ and } 0 \leqslant t \leqslant T\big\}$$

and

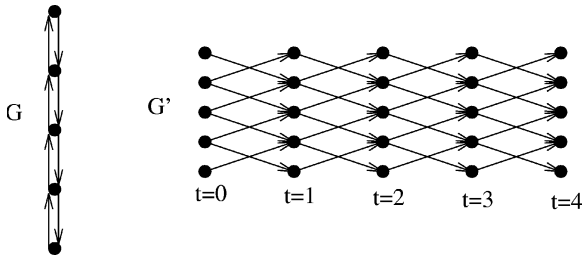

Fig. 1. A chain of five vertices and its corresponding multistage graph.

$$E' = \{((v,t),(w,t+1)) \mid w \in \mathit{neighbors}(v,G) \text{ and } 0 \leqslant t < T\}.$$

The edges in E' represent the communication that can take place between adjacent vertices of the interconnection network at any time.

Fig. 1 shows the resulting graph when the interconnection network is a chain of length 5. For routing permutations on a chain of five processors, an obvious upper bound of 4 routing steps applies.

Let $\mathit{tower}(G', v)$ be the set of vertices of graph G' (the constructed multistage graph) which correspond to vertex v in G. Formally,

$$\mathit{tower}(G', v) = \{(v,t) \mid 0 \leqslant t \leqslant T\}.$$

We can think of the stages of the multistage graph G' as representing time. In that sense, the route of any flit will be a directed path from a vertex in the flit's origin-tower to a vertex in the flits destination-tower. So, an off-line solution to a deflection worm routing problem can be seen as a collection of paths. However, the paths must satisfy several conditions which reflect the fact that we are routing worms and that only one flit can be transmitted along any communication link in a single step.

Definition. A *valid off-line solution of length* L for the deflection worm routing problem $R = (G, P)$ is a set of directed paths, one path for each flit of each worm, in the multistage graph G' of G, such that:

(i) the head of worm $p_i = (\mathit{orig}_i, \mathit{dest}_i) \in P$ travels from a vertex (orig_i, t') in $\mathit{tower}(G', \mathit{orig}_i)$ to vertex (dest_i, t'') in $\mathit{tower}(G', \mathit{dest}_i)$, $t' \leqslant t'' \leqslant L - k + 1$, where k is the number of flits in a worm,

(ii) if the jth flit of a worm, $1 \leqslant j < k$, travels from vertex (v,t) to vertex $(w,t+1)$,[2] then the $(j+1)$th flit of the worm travels from vertex $(v, t+1)$ to vertex $(w, t+2)$,

(iii) all paths are edge disjoint.

Given the above definition, the goal of an off-line deflection worm routing algorithm is to derive a collection of paths, one for each flit of each worm, such that they form a valid off-line solution of the smallest possible length L. The reduction of the routing problem to the derivation of a collection of edge disjoint paths on a multistage graph is referred as the *multistage off-line routing method* and it was introduced by Symvonis and Tidswell [7]. The method was originally used for deriving off-line solutions to single packet ($k = 1$) routing problems on meshes and tori. It was also used successfully in obtaining optimal off-line solutions for routing on trees [6].

3. Off-line optimal deflection worm routing

The off-line algorithm must describe for each worm the path that each of its flits takes. But, since we can deduce the movement of the whole worm from the movement of the head, only paths for the heads of the worms are necessary. Given a routing schedule of L steps and the fact that at most n^2/k worms might be moving at any step, we conclude that about Ln^2/k bits are necessary for the description of the routing schedule. Since $L = \Omega(kn)$, we expect that a routing schedule will require $\Omega(n^3)$ bits for its description.

However, the routing schedule that our algorithm will produce can be described with $O(n^2 \log(kn))$ bits. This is because the paths will be of a special form. More precisely, after each worm starts its routing it will move on a minimal path, first horizontally to its destination column and then vertically to its destination node. Since these *one-bend* paths can be deduced from each (*origin*, *destination*) pair, the only information that is required to describe the routing of a worm is the step in which

[2] In other words, the edge $((v,t),(w,t+1))$ belongs to the path assigned to the jth flit of the worm.

the worm starts moving. Thus, $O(n^2 \log(kn))$ bits suffice.

In the description of the algorithm, variable *start*$[p]$ contains the routing step in which worm p starts moving.

Algorithm *Off-line_mesh_routing*

1. Construct a multistage graph G' of $4kn$ stages for an $n \times n$ mesh as described in Section 2.
2. $G'_{\text{current}} = G'$
3. **while** there are more worms to be routed **do**
   (a) Let $p = (orig, dest)$ be the next worm to be routed.
   (b) $stage = 0$
   (c) $routed =$ **false**
   /* *routed* will become true when a set of edges has */
   /* been assigned to p. */
   (d) **while** (**not** *routed*) **do**
      (i) Let S be the set of edges of G' which are required in order to route worm p in such a way that the head departs from node $(orig, stage)$ and moves horizontally to the column destination and then vertically to *dest*.
      (ii) **if** $S \subseteq E(G'_{\text{current}})$
      then $E(G'_{\text{current}}) = E(G'_{\text{current}}) - S$
      $start[p] = stage$
      $routed =$ **true**
      else $stage = stage + 1$

Definition. When algorithm *Off-line_mesh_routing* is used to produce a routing schedule, we say that *worm p is delayed by worm q at step t* if worm q used during its routing the first edge which prevents (because it was removed from G'_{current}) worm p to start its routing (as described in the algorithm) at time t.

Lemma 1. *Consider two worms p and q which are routed with Algorithm* Off-line_mesh_routing. *Without loss of generality, assume that paths have already been assigned to the flits of worm q. Then, during the path assignment phase* (*step* 3(d)) *of the algorithm, worm q can delay worm p by at most $2k - 1$ steps.*

Proof. Worm p is delayed by worm q at time t if the first edge which is not present in G'_{current} but is required for the routing of p was used for the routing of a flit of q. Recall that worms move first horizontally to their destination column and then vertically to their destination node, that is, they move along minimal paths. We consider two cases:

(i) *An edge which was required for the routing of the head of p was used by q.* This corresponds to the head of p "bumping into" worm q. Since worm q consists of k flits, it can delay p by at most k steps.

(ii) *An edge which was required for the routing of a flit of the body of p was used by q.* The missing edge was used for the routing of the head of q. This corresponds to the head of q "bumping into" the body of p. By saying that the the head of q "bumped into" the body of p, we mean that, if the body of p consisted of a sufficiently smaller number of flits, then its routing could be completed. Since the body of p consists of $k - 1$ flits, p can be delayed by at most $k - 1$ steps.

From the two cases, we conclude that q can delay p by at most $2k - 1$ steps. □

Theorem 1. *Given an $n \times n$ mesh and a permutation π of its vertices that has to be routed using the deflection worm routing model where each worm consists of k flits, Algorithm* Off-line_mesh_routing *produces an optimal routing schedule of* $O(kn)$*-steps.*

Proof. The proof of the theorem follows from the property in Lemma 1 and the deadlock-free nature of the one-bend path algorithm. Consider an arbitrary worm p. When p is routed by Algorithm *Off-line_mesh_routing*, it can interfere with at most $2n - 2$ other worms, i.e., $n - 1$ worms which originate in the same row as p and $n - 1$ worms which are destined for the same column as p. From Lemma 1 we know that each of these worms can delay p by at most $2k - 1$ steps. Thus, p will start its routing after at most $(2n - 2)(2k - 1) + 1$ steps. Since it might be at most $2n - 2$ steps away from its destination, p's tail will reach its destination after at most $(2n-2)(2k-1)+1+(2n-2+k-1) < 4kn = O(kn)$ steps. A routing schedule of $O(kn)$ steps is asymptotically optimal. □

Analysis of Algorithm Off-line_mesh_routing

The algorithm requires memory to store the multistage graphs G', G'_{current}, set S and array *start*. This amounts to a total of $O(kn^3)$. A careful implementation of the algorithm can spare the use of G' and S (they were used in the presentation for clarity reasons).

For a worst case time analysis assume that each worm is delayed for $O(kn)$ steps. It takes $O(kn)$ steps in the worst case to realize that the worm is being delayed. This is because the missing edge from G'_{current} might be one of the last edges in the worm's (almost successful) potential route. Thus, a route will be assigned to each worm after $O(k^2n^2)$ time. Since there are n^2 worms to be routed, the algorithm will terminate after $O(k^2n^4)$ time. More careful implementation, talking into account the fact that a single worm can delay another worm at most $2k-1$ steps, can reduce the time complexity to $O(kn^4)$.

Variations of Algorithm Off-line_mesh_routing

Several versions of Algorithm *Off-line_mesh_routing* are possible and will produce similar results. What we presented is a very general version of it. We can refine the algorithm by specifying an order in which the worms will be routed. The arbitrary order used in the presentation resulted in an optimal algorithm. So any other ordering might only reduce the constant factor hidden in the *big-Oh* notation. When $k=1$ the presented algorithm results in a routing schedule of $4n-4$ steps. If the worms (or packets, since they have only 1 flit) are routed in a lexicographical order with respect to the pair $(hor, vert)$ where *hor* is the horizontal distance they have to travel and *vert* is the vertical one, a routing schedule of $3n-3$ steps will be produced (see [7] for details).

4. Conclusions

In this research note, we described an algorithm for off-line deflection worm routing of permutations in an $n \times n$ mesh in $O(kn)$-steps, where k is the number of flits in each worm. Even though the algorithm obeys the rules of deflection routing, it can be considered an "extreme" case of a deflection routing algorithm. This is because the presented algorithm avoids queuing by scheduling the time that each worm is sent on. However, the result of this paper suggests that it might be possible to get the same (optimal) performance with an on-line algorithm. Designing such an algorithm is an open problem.

References

[1] A. Bar-Noy, P. Raghavan, B. Schieber, H. Tamaki, Fast deflection routing for packets and worms, in: Proc. 12th Annual ACM Symposium on Principles of Distributed Computing (PODC 93), Ithaca, NY, August 1993, pp. 75–86.

[2] F. Makedon, A. Symvonis, Optimal algorithms for the many-to-one routing problem on two-dimensional meshes, Microprocessors and Microsystems 17 (6) (1993) 361–367.

[3] Newman, Schuster, Hot potato worm routing via store-and-forward packet routing, J. Parallel Distributed Comput. 30 (1) (1995) 76–84.

[4] C.P. Schnorr, A. Shamir, An optimal sorting algorithm for mesh connected computers, in: Proc. 18th Annual ACM Symposium on Theory of Computing, Berkeley, CA, ACM Press, New York, 1986, pp. 255–263.

[5] J.F. Sibeyn, M. Kaufmann, Deterministic $1-k$ routing on meshes, in: P. Enjalbert , E.W. Mayr, K.W. Wagner (Eds.), Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science, STACS 94 (Caen, France, February 1994), Lecture Notes in Comput. Sci., Vol. 775, Springer, Berlin, 1994, pp. 237–248.

[6] A. Symvonis, Optimal algorithms for packet routing on trees, in: Proc. 6th International Conference on Computing and Information (ICCI'94), Peterborough, Ontario, May 1994, pp. 144–161. Also TR 471, Basser Department of Computer Science, University of Sydney, September 1993.

[7] A. Symvonis, J. Tidswell, An empirical study of off-line permutation packet routing on 2-dimensional meshes based on the multistage routing method, IEEE Trans. Comput. 45 (5) (1996) 619–625.

[8] M. Tompa, Lecture notes on message routing in parallel machines, Technical Report 94-06-05, Department of Computer Science and Engineering, University of Washington, June 1994.