On Deflection Worm Routing on Meshes

Alan Roberts and Antonios Symvonis
Basser Department of Computer Science, University of Sydney,
N.S.W. 2006, Australia. {alanr,symvonis}@cs.su.oz.au

Abstract

In this paper, we consider the deflection worm routing problem on two dimensional $n \times n$ meshes. Our results include: (i) an off-line algorithm for routing permutations in $O(kn)$ steps, and (ii) a general method to obtain deflection worm routing algorithms from packet routing algorithms.

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*. In worm routing, the message is considered to be a worm; a sequence of k flits which, during the routing, follow the head of the worm which knows the destination address. If packets can be stored in intermediate nodes during the trip from their origin to their destination, the routing model is referred as *store-and-forward*. In a different model known as *deflection* (or *hot-potato*) routing, packets cannot be queued and are always moving until they reaches their destination.

In this paper, we concentrate on deflection worm routing on $n \times n$ meshes. Deflection worm routing on meshes was first examined by Bar-Noy, Schieber, Raghavan and Tamaki [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 randomised algorithms. Newman and Schuster [2] described a method to obtain deflection worm routing algorithms based on store-and-forward packet routing algorithms. However, the packet routing algorithms used in their method were restricted to use queues of at most four packets per processor. By employing the sorting algorithm of Schnorr and Shamir [3] 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 $1-h$ routing [4, 5], or $h-h$ 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 paper contributes to the literature of off-line and on-line deflection worm routing algorithms. We present an $O(kn)$-step off-line algorithm for routing permutations. The existence of such an algorithm was implied by the $O(kn)$-step randomised 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 [2]. (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.) We also generalise the method of Newman and Schuster [2] to allow it to use packet routing algorithms of queue-size $f(N)$, where $f(N)$ is a function of the side-length N of the mesh in which the packet routing algorithm is applied. The result dramatically increases the number of candidate packet routing algorithms that can be used in deriving deflection worm routing algorithms. The results presented in this paper appear in detail in [6].

2 Optimal Off-line Deflection Worm Routing

In our effort to derive an off-line solution for routing permutations using the deflection worm routing model, we use the multistage off-line routing method [7]. The method was originally used for deriving off-line solutions to packet routing problems on meshes and tori. It was also used successfully in obtaining optimal off-line solutions for routing on trees [8].

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. The elements in set P represent the m packets (worms) to be routed. Formally, $P = \{p_1, p_2, \ldots, p_m | \ p_i = (orig_i, dest_i), \ orig_i, dest_i \in V, \ 1 \leq i \leq m\}$.

Consider any routing problem $R = (G, P)$. Assume a upper bound of T routing steps for the problem under consideration. We construct a multistage directed graph $G' = (V', E')$ as follows: $V' = \{(v, t) | \ v \in V \text{ and } 0 \leq t \leq T\}$ and $E' = \{((v, t), (w, t+1)) \mid w \in neighbors(v, G) \text{ and } 0 \leq t < T\}$. The edges in E' represent the communication that can take place between adjacent vertices of the interconnection network at any time. Figure 1 shows the resulting graph when the interconnection network is a chain of length 5.

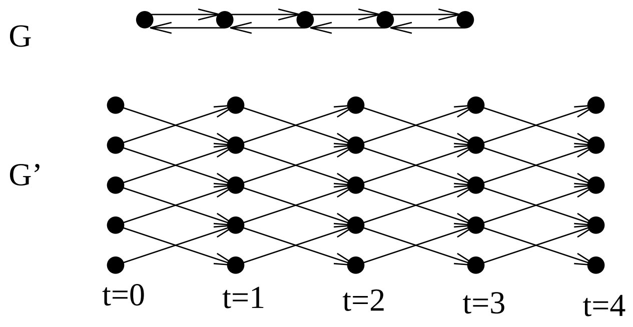

Figure 1: A chain of 5 vertices and its corresponding multistage graph.

Let $tower(G', v)$ be the set of vertices of graph G' (the constructed multistage graph) which correspond to vertex v in G. Formally, $tower(G', v) = \{(v, t) | \ v \in V, (v, t) \in V', 0 \leq t \leq T\}$.

We can think 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 flit's destination-tower. So, an off-line solution to a deflection worm routing problem can be seen as a collection of paths.

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 $p_i = (orig_i, dest_i) \in P$ travels from a vertex $(orig_i, t')$ in $tower(G', orig_i)$ to vertex $(dest_i, t'')$ in $tower(G', dest_i)$, $t' \leq t'' \leq L - k + 1$, where k is the number of flits in a worm
ii) if the j-th flit of a worm, $1 \leq j < k$, travels from vertex (v, t) to vertex $(w, t+1)$, 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 will be 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. 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 above.
 2. $G'_{current} = G'$
 3. while there are more worms to be routed do
     (a) Let $p = (orig, dest)$ be the next worm to be
         routed.
     (b) $stage = 1$
     (c) $routed$ = false /* $routed$ will become true
         when a set of paths 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 hor-
             izontally to the column destination and
             then vertically to $dest$.
         ii. if $S \subseteq G'_{current}$
               then
                 $E(G'_{current}) = E(G'_{current}) - S$
                 $start[p] = stage$
                 $routed$ = true
               else $stage = stage + 1$
```

Theorem 1 ([6]) *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 in $O(kn^4)$ time an optimal routing schedule of $O(kn)$-steps. Moreover, the routing schedule can be described with $O(n^2 \log(kn))$ bits.*

3 Routing on a Two-Dimensional $n \times n$ Mesh

We construct an efficient k-worm routing algorithm by treating each worm as though it were a packet and simulating the operation of a packet routing algorithm. Let $\mathcal{A}(N)$ be the packet routing algorithm (operating on an $N \times N$ mesh M_N) of which the operations we intent to simulate. We assume that $\mathcal{A}(N)$ completes the routing within $t_{\mathcal{A}}(N)$ steps and uses queues of size $f(N)$ packets. Algorithm $\mathcal{A}(N)$ is *suitable* for our method if the following assumption regarding the routing model is satisfied.

Assumption 1: On any given step all decisions made about the movement of any packet will be made locally by the node that currently stores the packets without considering the contents of any other node.

In the original work of Newman and Schuster [2], any suitable for simulation packet routing algorithm $\mathcal{A}(N)$ had to also satisfy:

Assumption 2: $\mathcal{A}(N)$ uses a queue-size of at most 4 packets.

Relaxing Assumption 2 results in an increase in the number of packet routing algorithms which are suitable for simulation. As we show, an $O((f(N)k)^{2.5}n)$-step algorithm can be derived from an $O(N)$-step packet routing algorithm which requires queue size of at most $f(N)$ packets and satisfies Assumption 1.

Let $\mathcal{A}(N)$ be a permutation packet routing algorithm which satisfies Assumption 1 and uses queues of size at most $f(N) < n$ packets. Choose N such that satisfies $N = n/(\sqrt{(f(N)+4)k}+1)$. We assume for simplicity that n, $f(N)$ and k are integers such that N, $\sqrt{k}$ and $\sqrt{f(N)+4}$ are integers. In addition, we assume that k is even. Note that this immediately implies that $\sqrt{(f(N)+4)k}$ is even.

For the purposes of the algorithm, we divide the mesh up into an $N \times N$ mesh of *supernodes*; each supernode being a sub-mesh of size $(\sqrt{(f(N)+4)k}+1) \times (\sqrt{(f(N)+4)k}+1)$. The rows and columns inside each supernode are numbered from 0 to $\sqrt{(f(N)+4)k}$ in the same way that the overall mesh is.

The algorithm is achieved in $(\sqrt{(f(N)+4)k}+1)^4$ *rounds.* During the (i,j,k,l)-th round we route all worms that originate at the (i,j) point of a supernode and are destined for point (k,l) of some supernode. Accordingly, at the beginning of each round, there is one worm generated inside each supernode. Each worm is then treated as though it were a packet. Decisions on sending worms from one supernode to another are made using a packet routing algorithm. The supernodes act like an $N \times N$ mesh of nodes with respect to this packet routing algorithm. Each supernode has a queue size of at most $f(N)$ packets. Each step of the packet routing algorithm is simulated by an underlying algorithm that determines the way in which the worms are moved about. We refer to a step of the packet routing algorithm as a *superstep.* Each round proceeds "superstep" by "superstep" until all worms have arrived at their destinations. The algorithm proceeds "round" by "round" until the whole permutation has been routed.

The high level description of the algorithm is identical to that of Newman and Schuster [2]. However, since we allow the use of a larger class of packet routing algorithms, we have to modify the structure of the supernode and to drastically refine the simulation of the superstep.

3.1 The Structure of Each Supernode

The layout of the sub-meshes that are used to simulate supernodes participating in the packet routing algorithms is shown in Figure 2. Each submesh contains several regions of importance. The regions include:

i) A vertical and a horizontal lane. The lanes are used as communication avenues between the supernode and its neighbores (supernodes).

ii) 4 buffers Each buffer is associated with one of the neighbores of the supernode. It is used to store the worm that is transmitted out of (received from) the (associated neighbor) supernode.

iii) 2 storage areas The storage areas are used to implement the queuesize that the packet routing algorithm uses. Note that the total size of the two storage areas is enough to store $f(N)$ worms, each consisting of k flits.

iv) 2 command processors north and south of the center of the supernode During the simulation, the worms that are stored in the supernode go through these two processors. Since the processors "see" all worms they can decide which of them will be transmitted during the next step.

See [6] for more details about the role of each region and for a justification for the use of two storage areas and two command processors.

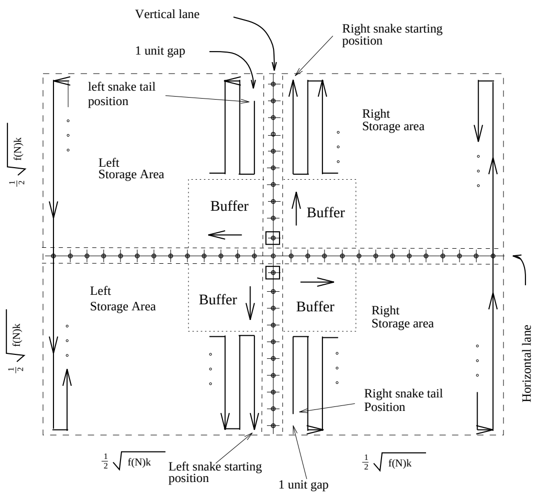

Figure 2: The sub-mesh that is used to simulate a supernode with respect to the packet routing algorithm $\mathcal{A}(n)$. Each buffer stores a single worm of length k.

3.2 The Simulation of a Superstep

Each superstep simulates the operation of a processor in the packet routing algorithm. An invariant of the simulation is that at the beginning of each superstep the buffers of a supernode are empty. This simply means that all packets currently at the supernode are held in the storage areas. Assuming that the precondition holds, a superstep can be considered to be a sequence of the phases:

A Superstep

- Output selection phase
- Extraction phase
- Transmission phase
- Queueing phase

At the beginning of each round, the worms that will be routed must be generated and placed in the storage snakes. This gives rise to another phase, the *creation phase.* See [6] for the specifics of the simulation of each phase. As a result of the simulation, we can state the following theorem:

Theorem 2 *Let $\mathcal{A}(N)$ be an $O(N)$-step permutation packet routing algorithm for M_N which uses queues of size $f(N)$ and satisfies Assumption 1. Then there is a hot-potato worm permutation routing algorithm $\widehat{\mathcal{A}(n)}$ for M_n, which routes k-worms in $O((f(N)k)^{2.5}n)$ steps, where N satisfies $N = \frac{n}{\sqrt{(f(N)+4)k}+1}$.*

References

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

[2] I. Newman and A. Schuster, "Hot potato worm routing via store-and-forward packet routing", in *First Israel Symposium on Theory of Computing and Systems (ISTCS'93)*, 1993, To appear in Journal of Parallel and Distributed Computing.

[3] C. P. Schnorr and A. Shamir, "An optimal sorting algorithm for mesh connected computers", in *Proceedings of the 18th Ann. ACM Symposium on Theory of Computing (Berkeley, CA)*. ACM, 1986, pp. 255–263, ACM Press.

[4] F. Makedon and A. Symvonis, "Optimal algorithms for the many-to-one routing problem on two-dimensional meshes", *Microprocessors and Microsystems*, vol. 17, no. 6, pp. 361–367, 1993.

[5] J.F. Sibeyn and M. Kaufmann, "Deterministic $1-k$ routing on meshes", in *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, STACS 94 (Caen, France, February 1994)*, P. Enjalbert, E. W. Mayr, and K. W. Wagner, Eds. 1994, LNCS 775, pp. 237–248, Springer-Verlag.

[6] A. Roberts and A. Symvonis, "On deflection worm routing on meshes", Tech. Rep. TR-490, Basser Dept of Computer Science, University of Sydney, Oct. 1994.

[7] A. Symvonis and J. Tidswell, "A new approach to off-line packet routing. Case study: 2-dimensional meshes", in *Proceedings of the 1992 DAGS/PC Symposium Dartmouth Institute for Advanced Graduate Studies in Parallel Computation, Hanover, NH, USA*, June 1992, pp. 84–93.

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

[9] A. Symvonis and J. Tidswell, "An empirical study of off-line permutation packet routing on 2-dimensional meshes based on the multistage routing method", Tech. Rep. TR-477, Basser Dept of Computer Science, University of Sydney, Feb. 1994, Submitted for journal publication.