Optimal Many-to-One Routing on the Mesh with Constant Queues*

(Extended Abstract)

Andrea Pietracaprina and Geppino Pucci

Dipartimento di Elettronica e Informatica, Università di Padova
Padova, Italy {andrea,geppo}@artemide.dei.unipd.it

Abstract. We present randomized and deterministic algorithms for many-to-one routing on an n-node two-dimensional mesh under the store-and-forward model. We consider a general instance of the problem, where each node is source (resp., destination) of ℓ (resp., k) packets, for arbitrary values of ℓ and k. All our algorithms run in optimal $O\left(\sqrt{\ell k n}\right)$ time and use queues of only constant size at each node to store packets in transit. The randomized algorithms, however, are simpler to implement. Our result closes a gap in the literature, where time-optimal algorithms using constant-size queues were known only for the special cases $\ell = 1$ and $\ell = k$.

1 Introduction

In this paper we study the routing of *many-to-one* message-sets on the mesh, where each node is the source and the destination of several messages [Lei92]. When the maximum number of messages originating at (resp., destined to) a node is ℓ (resp., k) the corresponding many-to-one routing instance is known in the literature as (ℓ, k)-*routing*. We will develop randomized and deterministic algorithms for (ℓ, k)-routing on an n-node square mesh. The performance of a store-and-forward algorithm is typically given by two key quantities: completion time, defined as the maximum delivery time of any packet to its destination, and maximum queue size (usually measured in packet units) needed at a node to store packets in transit. Our mesh algorithms turn out to be optimal with respect to both these metrics.

Routing under the store-and-forward model has been intensively studied over the last two decades [GH+98]. The first result on routing many-to-one message sets on an n-node square mesh is due to Makedon and Symovnis [MS93], who devised an optimal deterministic $O\left(\sqrt{kn}\right)$-time algorithm with constant queues for the special case of $(1, k)$-routing, where each node is the source of at most one packet. Subsequently, in [SK94], Sibeyn and Kaufmann proved an $\Omega\left(\sqrt{\ell k n}\right)$

* This work was supported, in part, by CNR and MURST of Italy under projects *Multicast Techniques with Applications to Robotics and Packet Routing* and *Algorithms for Large Data Sets: Science and Engineering.*

R. Sakellariou et al. (Eds.): Euro-Par 2001, LNCS 2150, pp. 645–650, 2001.

lower bound for general (ℓ, k)-routing (which holds for both randomized and deterministic algorithms) and obtained the first general, time-optimal deterministic algorithm, which however requires large queues of size $O(k)$. They also obtained a time-optimal randomized algorithm with constant queues and a more complex deterministic algorithm with similar performance for the case $\ell = k$. Their deterministic algorithm, however, works under the assumption that messages can be temporarily swapped out of the queues to be stored within the processors' internal memories, at the cost of a time penalty proportional to the length of the packet to be swapped out.

In this paper, we close the gap left open by the previous literature by devising time-optimal randomized and deterministic algorithms with constant queues for general (ℓ, k)-routing on the mesh. Both the algorithms implement a variant of the well-established idea of splitting the original message set into subsets of lower congestion that can then be routed independently within smaller submeshes [SK94]. However, the splitting is rather simple to achieve using randomization, while it requires a more complex and careful protocol to be accomplished deterministically.

2 Preliminaries

We make the reasonable assumption that messages departing from the same node have distinct destinations, which implies $\ell, k \leq n$. Every message is encapsulated into a distinct *packet* that consists of a header, containing the destination address, and a payload, containing the message itself. Each mesh node is provided with a *working queue* and an *internal queue*. The working queue is used during the routing to maintain packets in transit through the node, while the internal queue is used exclusively to hold the packets originating at or destined to the node. Hence, internal queues cannot be used for buffering purposes during the routing. The *queue size* of an algorithm is the maximum number of packets that any working queue must hold at any fixed time.

The mesh is synchronous and in one *step*, regarded as a unit of time, a node can perform a constant amount of local computation or one packet exchange along each of its four adjacent links. Also, extraction/injection of a packet from/into any internal queue takes constant time.

In our algorithms, we will make use of tessellations of the mesh into square submeshes of equal size. When the mesh is tessellated into s square submeshes of n/s nodes each, we call each such submesh an *s-tile*. Furthermore, we number the mesh nodes from 0 to $n-1$, according to the natural row-major indexing, and the s-tiles from 0 to $s-1$ according to a *hamiltonian indexing* so that s-tile i is adjacent to s-tile $(i+1)$ **mod** s, for every $0 \leq i < s$.

In what follows, all proofs are omitted for lack of space. Details will be provided in the full version of this abstract.

3 Randomized Algorithm

In this section we present a randomized algorithm for (ℓ, k)-routing on the mesh which attains optimal performance using constant queue size. Our algorithm builds upon the ideas employed in the (k, k)-routing algorithm developed by Sibeyn and Kaufmann [SK94], and extends their result to the general case of (ℓ, k)-routing with $\ell \neq k$. We assume that at the beginning of the routing the mesh nodes know the values ℓ and k. (This assumption can be easily removed by means of standard techniques [HPP01].)

For ease of presentation, we distinguish among the cases $\ell \leq k$ and $\ell > k$. Interestingly, the strategies in these two cases are somehow one the "mirror image" of the other.

3.1 (ℓ, k)-Routing with $\ell \leq k$

As in [SK94] the algorithm exploits an initial random ℓ-coloring of the packets, and delivers the packets of each color class in a separate stage. However, unlike the case $\ell = k$, the coloring does not reduce the problem to easily routable subproblems, and more sophisticated techniques are needed to deal with these subproblems.

The algorithm performs the following sequence of steps. Define $s = \sqrt{nk/\ell}$ and note that, since both ℓ and k are not larger than n, we have $s \geq k/\ell$.

1. Within each node, assign a *distinct* random *color* in $\{1, \ldots, \ell\}$ to each packet in the internal queue. Use the term *j-packet* to refer to a packet of color j. (Note that there are at most n j-packets, for each $j \in \{1, \ldots, \ell\}$.)
2. For each color j, $1 \leq j \leq \ell$, do the following:
   (a) Sort the j-packets in lexicographic order (destination s-tile, destination).
   (b) Reshuffle the j-packets so that a packet of rank r in the sorted sequence is sent to the node of index r **div** (k/ℓ) in the (k/ℓ)-tile of index r **mod** (k/ℓ).
   (c) Repeat k/ℓ times in each (k/ℓ)-tile:
      i. Route all j-packets with destinations within the (k/ℓ)-tile to their destination s-tile, so that each node of an s-tile receives roughly the same number of packets. (Note that s-tiles are contained within (k/ℓ)-tiles.)
      ii. Within each s-tile move the j-packets along a hamiltonian cycle of the tile's nodes, thus letting each packet reach its destination.
      iii. Perform a blockwise shift of all unrouted j-packets to bring them to the same position within the next (k/ℓ)-tile in the hamiltonian indexing of the (k/ℓ)-tiles.

The analysis of the algorithm relies on the following lemma and theorem.

Lemma 1. *The coloring performed in Step 1 guarantees that for every $j \in \{1, \ldots, \ell\}$ the number of j-packets destined to the same s-tile is $O((n/s)(k/\ell))$, with high probability.*

Theorem 1. *For any $\ell \leq k$, the above algorithm performs (ℓ, k)-routing in optimal $O\left(\sqrt{\ell k n}\right)$ time using constant queue size, with high probability.*

3.2 (ℓ, k)-Routing with $\ell > k$

As mentioned before, the algorithm for the case $\ell > k$ can somehow be seen as a backward run of the previous algorithm. However, some slight modifications are needed. The algorithm consists of the following sequence of steps. Define $s = \sqrt{n\ell/k} \geq \ell/k$ and $s' = \sqrt{nk/\ell}$.

1. Within each node, assign a random *color* in $\{1, \ldots, k\}$ to each packet in the internal queue. Use the term *j-packet* to refer to a packet of color j. (Note that, since $k \leq \ell$, more than one packet in a node may be assigned the same color.)
2. For each color j, $1 \leq j \leq k$, do the following:
   (a) Rank all j-packets so that the ranks assigned to the j-packets originating from the same s-tile form an interval of consecutive integers.
   (b) For $0 \leq i < \ell/k$ do the following within each (ℓ/k)-tile T:
      i. Let T be the (ℓ/k)-tile of index u in the hamiltonian indexing of the (ℓ/k)-tiles. From each s-tile contained in T, inject all j-packets whose rank r is such that $(u - r)$ **mod** $(\ell/k) = i$, and distribute such packets among the nodes of the s-tile. The injection is accomplished by viewing each s-tile as a linear array of n/s nodes and applying a straightforward balancing algorithm [Lei92].
      ii. Reshuffle all j-packets currently residing in the working queues of the nodes of T, so that they are evenly distributed among such queues.
      iii. Perform a blockwise shift of all j-packets to bring them to the same position within the next (ℓ/k)-tile in the hamiltonian indexing of the (ℓ/k)-tiles.
   (c) Route all j-packets to their destination s'-tile so that each node of an s'-tile receives roughly the same number of packets.
   (d) Within each s'-tile move the packets along a hamiltonian cycle of the nodes of the tile, thus letting each packet reach its destination.

The analysis of the algorithm relies on the following lemma and theorem.

Lemma 2. *The coloring performed in Step 1 guarantees that, with high probability, for every color $j \in \{1, \ldots, k\}$, every s-tile U, and every s'-tile U', the following properties hold: (i) The total number of j-packets is $O(n)$; (ii) The number of j-packets with sources in U is $O((n/s)(\ell/k))$; (iii) The number of j-packets with destinations in U' is $O(n/s')$.*

Theorem 2. *For any $\ell > k$, the above algorithm performs (ℓ, k)-routing in optimal $O\left(\sqrt{\ell k n}\right)$ time using constant queue size, with high probability.*

4 Deterministic Algorithm

In the algorithms presented in the previous section, randomization is employed exclusively to assign colors to the packets, so to partition them into subsets characterized by lower congestion at source or destination tiles of suitable size. Therefore, in order to obtain a deterministic algorithm we must adopt a (more sophisticated) coloring strategy that provides similar guarantees in the worst case. The required modifications to the algorithms are described below.

Let us first consider the case $\ell \leq k$. The coloring performed in Step 1 of the randomized algorithm for this case can be substituted with the following computation. Let $s = \sqrt{nk/\ell}$.

1. In parallel for each s-tile T, rank the packets destined to T with consecutive integers ensuring that packets whose sources are in consecutive nodes of the mesh receive consecutive ranks. Assign color j to every packet whose rank r is such that r **mod** $\ell = j$, with $0 \leq j < \ell$. Call j-packets the packets of color j.

Step 1 can be accomplished in $O(s + \sqrt{n})$ time by running s pipelined prefix operations on the entire mesh, where each prefix ranks the packets destined to a distinct s-tile.

It is easy to see that, for every j, there are $O((n/s)(k/\ell))$ j-packets with destination in the same s-tile, and there are $O(s)$ j-packets originating at the nodes of any stripe of $\lceil s/\sqrt{n} \rceil$ rows of the mesh. However, the coloring does not guarantee that a node has only $O(1)$ j-packets. Therefore, the sorting step (Step 2.(a)) of the randomized algorithm must be modified as follows.

2.(a).i Evenly distribute the j-packets within each stripe of $\lceil s/\sqrt{n} \rceil$ consecutive rows.

2.(a).ii Sort the j-packets in lexicographic order (destination s-tile, destination).

Step 2.(a).i can be accomplished as a balancing of $O(s)$ packets in a linear array of $O(s)$ nodes in $O(s)$ time [Lei92]. The rest of the algorithm is identical to the randomized one.

Consider now the case $\ell > k$. We modify the randomized algorithm for this case by substituting the coloring performed in Step 1 with the following computation. Let $s = \sqrt{n\ell/k} \geq \ell/k$ and $s' = \sqrt{nk/\ell}$.

1. In parallel for each s'-tile T, rank the packets destined to T with consecutive integers ensuring that packets whose sources are in the same s-tile receive consecutive ranks. Assign color j to every packet whose rank r is such that r **mod** $k = j$, with $0 \leq j < k$. Call j-packets the packets of color j.

It is easy the see that the above coloring, which can be performed with techniques aking to those used for the case $\ell \leq k$, guarantees that the three properties stated in Lemma 2 hold in the worst case.

Theorem 3. *Any instance of (ℓ, k)-routing can be performed in optimal $O\left(\sqrt{\ell kn}\right)$ time in the worst case using constant queue size.*

References

FRU96. S. Felperin, P. Raghavan, and E. Upfal. A theory of wormhole routing in parallel computers. *IEEE Trans. on Computers*, C-45(6):704–713, June 1996.

GH+98. M. D. Grammatikakis, D. F. Hsu, , M. Kraetzel, and J. F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, November 1998. 645

HR90. T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33(6):305–308, February 1990.

HPP01. K. T. Herley, A. Pietracaprina, and G. Pucci. One-to-many routing on the mesh. In *Proc. of the 13th Symp. on Parallel Algorithms and Architectures*, June 2001. To appear. 647

KK79. P. Kermani and L. Kleinrock. Virtual cut through: a new computer communication switching technique. *Computer Networks*, 3(4):267–286, April 1979.

Lei92. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992. 645, 648, 649

MS93. F. Makedon and A. Symvonis. Optimal algorithms for the many-to-one routing problem on two-dimensional meshes. *Microprocessors and Microsystems*, 17:361–367, 1993. 645

NS95. I. Newman and A. Schuster. Hot-potato worm routing via store-and-forward packet routing. *Journal of Parallel and Distributed Computing*, 30(1):76–84, January 1995.

SK94. J. F. Sibeyn and M. Kaufmann. Deterministic 1-k routing on meshes, with application to hot-potato worm-hole routing. In *Proc. of the 11th Symp. on Theoretical Aspects of Computer Science*, pages 237–248, March 1994. 645, 646, 647