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

Andrea Pietracaprina [a] Geppino Pucci [a,*]

[a] *Dip. di Ingegneria dell'Informazione, Università di Padova, Padova, Italy*

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 of packet routing. We consider the general instance of many-to-one routing where each node is the 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 $\ell = k$.

Key words: Parallel algorithms, Interconnection networks, Mesh, Packet routing

1 Introduction

Efficient interprocessor communication is one of the crucial characteristics of a parallel machine. When the machine's architecture consists of a set of processor/memory pairs (*nodes*) connected in a network topology of point-to-point links, interprocessor communication is typically performed via *packet switching*. In packet swicthing, each message originating at a node s is encapsulated in a packet and is shipped to its intended destination d by letting it travel along a path from s to d. One of the most thoroughly studied models of packet

⋆ This research was supported, in part, by MURST of Italy under the PNR project *ALGO-NEXT: ALGOrithms for the NEXT Generation Internet and the Web*. An abstract of this work was presented at the *7th European Conference on Parallel Computing – EUROPAR 2001*, Manchester UK, August 2001

* **Corresponding Author**: Dip. di Ingegneria dell'Informazione,Università di Padova, Via Gradenigo 6/B, 35131 Padova, Italy. Tel. +39 049827951, Fax: +39 049827799

Email addresses: andrea.pietracaprina@unipd.it (Andrea Pietracaprina), geppino.pucci@unipd.it (Geppino Pucci).

switching is *store-and-forward* [2], which requires packets to "hop" between neighboring nodes as indivisible units. When needed, queues are provided at the nodes for storing packets in transit that compete for transmission over the same link.

Other popular packet switching models are *wormhole routing* [1], where a packet is sent along its path as a contiguous sequence of small fragments called *flits*, and each node can buffer only a few flits of a packet; and *virtual cut-through* [4], which is similar to wormhole routing, except that the queues at the nodes are large enough to contain a small number of whole packets, rather than flits.

In this paper we focus on the store-and-forward model. The performance of a store-and-forward routing algorithm is stated in terms of two key quantities. The first is the algorithm's running time, defined as the maximum delivery time of a packet to its destination. The second quantity is the maximum queue size (usually measured in packet units) needed at a node to store packets in transit. Keeping the queue size small is a compelling practical requirement, since these queues are usually built in hardware to enable fast access by the routing circuitry of a node. In particular, routing algorithms working with constant queue capacity are highly desirable, since such algorithms also guarantee the scalability of the parallel machine, which can be made larger without changing the structure of a single node. Last, but not least, an important by-product of store-and-forward routing algorithms with constant queue size is that they are amenable to efficient translation into the more effective but more stringent wormhole and virtual cut-through models [7,1].

Customarily, a routing instance is usually described in terms of a *message set*, containing all the messages to be routed concurrently to their destinations. We distinguish between *unicast* message sets, where each message is associated to a single destination, and *multicast* message sets, where each message is to be delivered to several destinations. In this paper we study the most general routing problem for the case of unicast message sets under the store-and-forward model, namely, the routing of *many-to-one* message-sets, where each node is the source and the destination of several messages. 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 for an n-node square mesh. Such an interconnection has been widely studied in the literature due to its simplicity, regularity and scalability [5]. Our algorithms are time-optimal and work with constant queue capacity at each node, hence optimizing both the performance measures discussed above.

Routing under the store-and-forward model has been intensively studied over the last two decades. We will not attempt to summarize the entire literature on this problem here but only quote those results that directly relate to our work, and refer the interested reader to [2] (and the over 300 references therein) for

a recent and comprehensive summary of previous work on this topic.

The first result on routing many-to-one message sets on an n-node mesh are due to Makedon and Symovnis [6], 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 [8], Sibeyn and Kaufmann proved an $\Omega\left(\sqrt{\ell kn}\right)$ 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 [8]. However, the splitting is rather simple to achieve using randomization, while it requires a more complex and careful protocol to be accomplished deterministically.

The rest of the paper is organized as follows. Section 2 provides a detailed description of the machine and the routing model. Section 3 describes the randomized algorithm and Section 4 its deterministic counterpart.

2 Preliminaries

As stated in the introduction, an instance of the general many-to-one routing problem is an (ℓ, k)-*routing*, that is a unicast message set where each processor is the source of at most ℓ messages and the destination of at most k messages. 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.

Our topology of reference is the n-node mesh, where nodes are connected to form a $\sqrt{n} \times \sqrt{n}$ square grid. We adopt the following machine model defined in [8]. Each 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 the node prior to their injection into the network, and the packets destined to the node after completion of their journey. Hence,

internal queues cannot be used for buffering purposes during the routing. The *queue size* of an algorithm is the capacity required of each working queue, i.e., the maximum number of packets that the 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 one of the following activities:

- execute a constant number of elementary operations on packets held in the working queues or still residing in the internal queues (i.e., not yet injected);
- transmit/receive one packet along each of its four incident (bidirectional) links;
- transfer a packet originating at (resp., destined to) the node from the internal (resp., working) queue to the working (resp., internal) queue.

An easy bandwidth-based argument shows that for every ℓ and k there exists an (ℓ, k)-routing problem for which $\Omega\left(\sqrt{\ell k n}\right)$ steps are required to deliver all packets to their destinations [8]. The objective of the paper is to develop time-optimal algorithms for (ℓ, k)-routing, using constant queue size.

In the algorithms, we will make use of tessellations of the mesh into square submeshes of equal size. For convenience, we assume that n is an even power of two. When the mesh is tessellated into t square submeshes of n/t nodes each (where t is itself an even power of two [1]), we will call each such submesh a *t-tile*. Furthermore, we number the mesh nodes from 0 to $n-1$, according to the natural row-major indexing, and we number the t-tiles from 0 to $t-1$ according to a *hamiltonian indexing* so that t-tile i is adjacent to t-tile $(i+1)$ **mod** t, for every $0 \leq i < t$ (observe that such indexing exists since t is even).

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 [8], 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. The assumption can be easily removed, without affecting the performance and the queue size, as follows: ℓ can be precomputed via a standard prefix operation; as for k, we run the algorithm for geometrically increasing *guesses* k_i of k, aborting a run (by removing all packets in the working queues) when either the time goes beyond an allotted number of steps proportional to $\sqrt{\ell k_i n}$, or the queue capacity at a node is exceeded. Clearly, the overall running time is proportional to the time

[1] Throughout the paper, tessellations with different values of t are used, where t is a function of the parameters of the routing instance. In case the resulting value of t were not an even power of two, the tessellation is obtained by choosing the smallest even power of two t' larger than t, i.e., $t' = 4^{\lceil \log_4 t \rceil}$

taken by the last run (plus an additive, lower-order term which accounts for resynching the mesh nodes after each aborted run), which is optimal.

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 [8] 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 $k/\ell \leq s \leq n$.

(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.
Comment: Since each node generates at most one j-packet, there are at most n j-packets overall, 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) Route 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.
      Comment: Since $s \geq k/\ell$, 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.

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.*

PROOF. Consider an arbitrary s-tile T and let $p(i,j)$ denote the probability that node i has a j-packet with destination in T, for $0 \leq i < n$ and $1 \leq j \leq \ell$. Clearly, if node i is the source of $x \leq \ell$ packets with destinations within T, then $p(i,j) = x/\ell$. Since there are at most kn/s packets with destinations in T overall, we have that $\sum_{i=0}^{n-1} p(i,j) \leq (n/s)(k/\ell)$. Then, by Chernoff's bound [3] we have that, for every $\epsilon > 0$, the probability that there are more than $(1+\epsilon)(n/s)(k/\ell)$ j-packets with destinations in T is bounded above by

$$\left(\frac{e}{1+\epsilon}\right)^{(1+\epsilon)(n/s)(k/\ell)} = \left(\frac{e}{1+\epsilon}\right)^{(1+\epsilon)\sqrt{nk/\ell}}$$

which, for $\epsilon > e-1$, is exponentially small in n. The lemma follows by applying the union bound over all colors and over all s-tiles.

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

PROOF. Consider the routing of j-packets. Lemma 1 implies that, with high probability, after the routing performed in Step 2.(b), in each (k/ℓ)-tile there are $O(n/s)$ j-packets with destination in the same s-tile. Consequently, in every execution of Step 2.(c).i, every s-tile receives a number of packets proportional to its size. This fact, together with the observation that at most one j-packet departs from a node, shows that constant queue size at each node suffices to hold all packets going through a node. Let us now evaluate the overall running time of the algorithm. Step 1 can be performed in $O(\ell)$ local steps at each node. For every color j, Steps 2.(a) and 2.(b) are executed via standard sorting, prefix and permutation routing that require $O(\sqrt{n})$ time with constant queues. Finally, each of the k/ℓ iterations of Step 2.(c) entails the execution of $(O(1), O(1))$-routing within (k/ℓ)-tiles (Step 2.(c).i) or between adjacent (k/ℓ)-tiles (Step 2.(c).iii), and complete tours within s-tiles (Step 2.(c).ii), which take $O\left(\sqrt{n\ell/k} + n/s\right) = O\left(\sqrt{n\ell/k}\right)$ time altogether, with constant queues. Hence the overall running time is

$$O\left(\ell\left(\sqrt{n} + \frac{k}{\ell}\sqrt{\frac{n\ell}{k}}\right)\right) = O\left(\sqrt{\ell kn}\right),$$

which is optimal.

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. **Comment:** 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 evenly distribute such packets among the nodes of the s-tile.
      (ii) Balance 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 technical lemma.

Lemma 3 *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:*

- *The total number of j-packets is $O(n)$.*
- *The number of j-packets with sources in U is $O((n/s)(\ell/k))$;*
- *The number of j-packets with destinations in U' is $O(n/s')$.*

PROOF. Fix a color j, an s-tile U and an s'-tile U'. The three properties are easily shown by applying the Chernoff's bound, as in Lemma 1, since packets are colored independently and the claimed values for the number of j-packets, the number of j-packets with sources in U, and the number of j-packets with destinations in U', are not smaller than the respective averages, which, in turn, are all $\Omega(\sqrt{n})$. The lemma follows by applying the union bound over all colors, all s-tiles, and all s'-tiles.

We have:

Theorem 4 *For any $\ell > k$, the above algorithm performs (ℓ, k)-routing in optimal $O\left(\sqrt{\ell kn}\right)$ time using constant queue size, with high probability.*

PROOF. The initial coloring can be performed in $O(\ell)$ local computation steps at each node. Consider now the routing of j-packets. Step 2.(a) is easily executed in $O(\sqrt{n})$ time with constant queue size using standard primitives. Next, we observe that at the end of Step 2.(b) the working queues of the nodes in the (ℓ/k)-tile T of index u contain all those j-packets that, in Step 2.(a), have been assigned a rank r such that r **mod** $(\ell/k) = u$. Moreover, such packets are evenly distributed among these queues. Hence, by the first property stated in Lemma 3, each such queue contains $O(1)$ packets. Also, the second property stated in the lemma and the ranking ensure that in each iteration of Step 2.(b) $O(n/s)$ j-packets are injected from each s-tile. These packets can be injected in $O(n/s)$ time with constant queues by considering the s-tile as a ring of n/s nodes and applying the greedy balancing algorithm for the ring topology [5]. Therefore, Step 2.(b) can be executed in time $O\left((\ell/k)(n/s + \sqrt{nk/\ell})\right) = O\left(\sqrt{n\ell/k}\right)$ using standard primitives, and requires only constant queue size. The third property stated in Lemma 3 implies that constant queue size is also sufficient for Steps 2.(c) and 2.(d). Step 2.(c) requires prefix, sorting and $(O(1), O(1))$-routing in the whole mesh, thus taking $O(\sqrt{n})$ time, while Step 2.(d) takes time $O(n/s') = O\left(\sqrt{n\ell/k}\right)$. Therefore, the entire algorithm runs in time

$$O\left(\ell + k\sqrt{n\ell/k}\right) = O\left(\sqrt{\ell k n}\right),$$

which is optimal, and requires only constant queue size.

4 Deterministic algorithm

Note that 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.

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).

The rest of the algorithm is identical to the randomized one.

Theorem 5 *For any $\ell \leq k$, (ℓ, k)-routing can be performed in optimal $O\left(\sqrt{\ell k n}\right)$ time in the worst case using constant queue size.*

PROOF. The deterministic coloring performed in Step 1 can be accomplished by a pipelined execution of s prefix computations on the mesh, where each prefix ranks the packets destined to a distinct s-tile. Assume for now that each node maintains s local variables, one for each prefix. For $0 \leq j < \sqrt{n}$ and $0 \leq i < s$, in row j the i-th prefix starts skewed at time $i + j + 1$ at the first node of the row, which initializes a counter with the number of packets locally held and destined to the s-tile of index i. Then, the counter is sent eastwards along the row. When the counter reaches a node it is first stored in the appropriate local variable, then incremented by the number of local packets destined to s-tile i, and finally sent eastwards. The last node in each row $j > 0$ receives a counter from the northern link, sends it westwards, adds it to the counter of its row and dispatches it southwards. While the counters move westwards along the rows, the local variables are updated to hold the final prefix values.

The last prefix will terminate after $O(s + \sqrt{n})$ time and there will never be more than two counters in each working queue of a mesh node. Moreover, rather than having s local variables in each node to store the values of the s prefixes, in $O(\ell \log \ell)$ time we can set up a data structure whose size does not exceed the aggregate size of the headers of the packets originating at the node, which stores the relevant rank values determined by the prefix computation. Thus, the time required by Step 1 is $O(\ell \log \ell + s + \sqrt{n}) = O(s)$ with constant queue size.

Finally, the distribution performed in Step 2.(a).i can be regarded as a balancing of $O(s)$ packets in a ring of $O(s)$ nodes, hence it can be accomplished in $O(s)$ time using standard techniques [5].

The analysis of the rest of the algorithm is identical to the one of the randomized algorithm, and the theorem follows.

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 guarantees that the three properties stated in Lemma 3 hold in the worst case. The coloring can be performed in time $O(s' + \sqrt{n}) = O(\sqrt{n})$ by using techniques akin to those described in the proof of Theorem 5. We have:

Theorem 6 *For any $\ell > k$, (ℓ, k)-routing can be performed in optimal $O\left(\sqrt{\ell kn}\right)$ time in the worst case using constant queue size.*

References

[1] S. Felperin, P. Raghavan, E. Upfal, A theory of wormhole routing in parallel computers, IEEE Trans. on Computers C-45 (6) (1996) 704–713.

[2] M. Grammatikakis, D. Hsu, , M. Kraetzel, J. Sibeyn, Packet routing in fixed-connection networks: A survey, Journal of Parallel and Distributed Computing 54 (2) (1998) 77–132.

[3] T. Hagerup, C. Rüb, A guided tour of Chernoff bounds, Information Processing Letters 33 (6) (1990) 305–308.

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

[5] F. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes, Morgan Kaufmann, San Mateo, CA, 1992.

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

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

[8] J. Sibeyn, 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, LNCS 775, 1994, pp. 237–248.