Optimal Permutation Routing on Mesh Networks

Ignasi Sau, *Email: Ignasi.Sauvalls@sophia.inria.fr,*
Mascotte Project, CNRS/UNSA/INRIA 2004 route des Lucioles - B.P. 93 F-06902 Sophia-Antipolis Cedex, France.

Janez Žerovnik, *Email: janez.zerovnik@imfm.uni-lj.si,*
IMFM, Jadranska 19, SI-1000 Ljubljana, Slovenia and University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia.

Keywords: mesh networks, hexagonal networks, permutation routing, shortest path, distributed algorithm, communication networks.

1 Introduction

Permutation routing is used as one of the standard tests of routing algorithms. In the permutation routing problem, each processor is the origin of at most one packet and the destination of no more than one packet. The goal is to minimize the number of time steps required to route all packets to their respective destinations. Wireless mesh networks are based on plane tessellations that divide the area into cells and give rise to triangular, square, and hexagonal grids. In this paper we study permutation routing algorithms that work on finite convex subgraphs of basic grids, under the store-and-forward Δ-port model. We consider algorithms implemented independently at each node, without assuming any global knowledge about the network. I.e., distributed algorithms.

We describe optimal distributed permutation routing algorithms for subgraphs of triangular and square grids that need ℓ_{max} (the maximum over the length of the shortest path of all packets) routing steps, and show that there is no such algorithm on the hexagonal grids. Furthermore, we show that these algorithms are oblivious and translation invariant.

2 Permutation Routing

The packet-routing problem on any interconnection network is essentially important. This problem involves how to transfer the right data to the right place within a reasonable amount of time. To measure the routing capability of an interconnection network, the *partial permutation routing* (PPR for short) problem is usually used as the metric. In the PPR problem, each processor is the origin of at most one packet and the destination of no more than one packet. Formally, we are given a general graph (that models a communication network), a subset V of nodes, and a permutation π that acts on this subset $\pi : V \to V$. Each node $u \in V$ wants to send a message to the node $\pi(u)$. Thus, we have $|V|$ pairs of communicating nodes and $|V|$ messages to be delivered simultaneously. All the edges and nodes of the *host* graph (the hexagonal graph if the network is a triangular grid) can be crossed by the packets. The goal is to minimize the number of time steps required to route all packets to their respective destinations.

The special case in which only one node has a packet to send is known as *2-terminal routing* problem. In this case, optimal algorithms have been found, for instance, in 2-jump circulant graphs [11] and hexagonal networks [10]. Under the store-and-forward Δ-port model [4], at each step a packet can either stay or move to an adjacent node by crossing a link, but no link (recall that in the full-duplex case, there are 2 links between 2 adjacent nodes) can be crossed by two packets at the same step. A node can send or receive packets through all its incident edges at the same time. Cohabitation of multiple packets at the same node is allowed. Thus, a queue is required for each outgoing edge at each node.

The permutation routing problem has been studied in a wide diversity of scenarios, such as Mobile Ad Hoc Networks [8], Cube-Connected Cycle Networks [7], Wireless and Radio Networks [2], All-Optical Networks [9] and Reconfigurable Meshes [1], among others.

An optimal algorithm for permutation routing was found in 2-circulant graphs [5] for full-duplex model. Routing algorithms for 2-circulants with half-duplex links are studied in [3]. In this paper we present the first optimal algorithm for permutation routing in full-duplex hexagonal networks. More details about this algorithm can be found in [12]. Finally, we also sketch a proof of optimality of a similar algorithm for square grid networks and briefly describe a 2-approximation algorithm for hexagonal grid networks.

3 Network Topologies

It is well known that there only exist three possible tessellations of the plane into regular polygons [10]. Tessellation of the plane with hexagons may be considered most natural because the cells have optimal diameter to area ratio. If centers of neighboring cells are connected, we obtain a triangular grid. This network can also be obtained from the basic 4-mesh by adding NE to SW edges, which is called a 6-mesh in [14].

Note that here is an ambiguity in the notation, because a triangular grid is often called a hexagonal network, whereas a hexagonal grid may be referred as a honeycomb network [13], as it is illustrated in Figure 1a. Hexagonal networks are finite subgraphs of the triangular grid. Finally, a tessellation of the plane with squares is called a square grid.

Two-dimensional meshes are doubtless among the most studied topologies for computer networks. Here we study *convex* subgraphs (that is, the subgraphs that contain all shortest paths between all pairs of nodes) of the triangular grid.

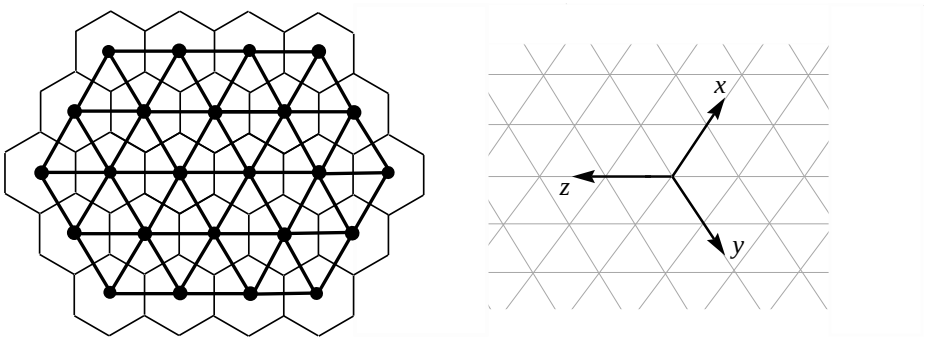

Figure 1: **a)** Hexagonal network ($\triangle$) and hexagonal tessellation ($\hexagon$). **b)** Axis used in a hexagonal network

More precisely, we deal with a hexagonal mesh network with full-duplex links. That is, an edge of the network can be crossed by two messages simultaneously, one in each direction. In other words, each edge between 2 nodes u and v is made of 2 independent arcs, $\{uv\}$ and $\{vu\}$.

If the network is not full-duplex, it is easy to construct a 2 factor approximation algorithm from an optimal algorithm for the full-duplex case by introducing *odd-even* steps, as explained for example in [3].

4 Algorithm for Triangular Grid

4.1 Preliminaries

In this work we strongly use some fundamental results of [10], where the problem of routing a *single* message, i.e. 2-terminal routing, through a hexagonal communication network is solved. The first idea that will be of our interest (in fact, this was firstly introduced in [13]) is the representation of the address of the nodes on a basis consisting of three unitary vectors ***i***, ***j***, ***k*** on the directions of three axis x, y, z with a 120 degree angle among them (see Figure 1b). Thus, each node is labeled with an address expressed on this basis $\{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$ with respect to any fixed arbitrary node. This address is not unique, but it can be easily proved [10] that, being (a, b, c) and (a', b', c') the addresses of two $D - S$ pairs, then $(a, b, c) = (a', b', c')$ if and only if there exists an integer d such that $a' = a + d$, $b' = b + d$, and $c' = c + d$. A relative address $D - S = (a, b, c)$ is said to be of the *shortest path form* if there is a path from node S to node D, consisting of a units of vector ***i***, b units of vector ***j*** and c units of vector ***k***, and this path has the shortest length over all paths going from S to D. The next result simplifies extraordinarily the routing on hexagonal networks.

Theorem 1 ([10]) *A relative address $D - S = (a, b, c)$ is of the shortest path form if and only if at least one component is zero (that is, $abc = 0$), and any two components do not have the same sign (that is, $ab \leq 0,\ ac \leq 0,\ and\ bc \leq 0$).*
Furthermore, the shortest path form is one of $(0, b - a, c - a)$, $(a - b, 0, c - b)$, *and* $(a - c, b - c, 0)$.

Given a packet p and its relative address (a, b, c) of the shortest path form, denote by ℓ_p the length of this shortest path, and by ℓ_{max} the maximum over the length of the shortest paths of all packets:

$$\ell_p := |a| + |b| + |c|\,, \qquad \ell_{max} := \max_p(\ell_p)$$

Observe that, since $|V| < \infty$, this maximum is indeed achieved (possibly by several messages). Trivially, ℓ_{max} is a lower bound for all algorithms. Two packets p and p' are *in conflict* or, simply, *meet*, if they are simultaneously in the same outgoing queue at the same node of the network. If two (or more) packets meet, only one of them will be able to move on the next step of the algorithm. Denote by w_p^i the number of steps waited by packet p until the end of the i*th* step of the algorithm. Call w_p^i the *waiting time of* p or, simply, the *delay of* p. Given a packet p and its delay w_p^i, w_p^i is an *allowed delay* if $\ell_p + w_p^i \leq \ell_{max}$. Similarly, w_p^i is an *additional* (or *forbidden*) *delay* if $\ell_p + w_p^i > \ell_{max}$. Finally, a packet p is *saturated* at the end of step i if $w_p^i = \ell_{max} - \ell_p$. The idea is that if a packet p is saturated, it cannot wait anymore unless the algorithm becomes not optimal. See Figure 2 for a logical model of a packet representing these concepts, used only for analysis.

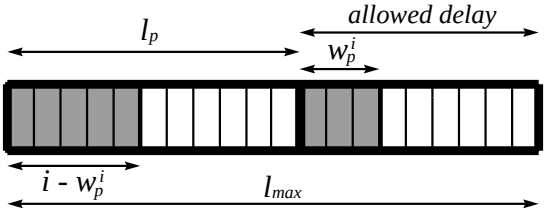

Figure 2: Logical packet model used on the permutation routing algorithm

4.2 Description of the algorithm

Our distributed algorithm, namely $\mathcal{A}$, can be now simply described as follows. At each node u of the network:

Initially, if there is a packet at this node, the preprocessing phase consists just in computing the relative address $D - S$ of the message in the shortest path form, and add this information to the message constituting the packet to be sent. Recall that because of Theorem 1, $D-S$ can have no more than one negative component. At each step, when a packet is received at u, its relative address is updated. Then, the transmission phase works as follows:

a) If there are packets with negative components, send them immediately along the direction of this component.

b) If not, for each outgoing edge order the packets according to decreasing number of remaining steps (that is, $\ell_p - i + w_p^i$) and send the first packet of each queue.

4.3 Correctness, running time and optimality

Algorithm $\mathcal{A}$ is really cheap in terms of computational cost, since the only involved operations are integer addition and comparison among the lengths of the addresses of the packets at each node. Let us briefly discuss the correctness and give the main ideas of the proof of optimality.

The rules given by Algorithm $\mathcal{A}$ define two directions of movement for each packet. That is, first of all a packet moves along the direction of its negative component, and then along the positive one. Obviously, if a packet has only positive component, it always moves along this direction. The first key observation is that packets can only wait, possibly, during their last direction. That is because if two packets meet when their first direction is not finished yet, it is easy to check that they must have the same origin node, a contradiction. Thus, in **a)** there can be at most one packet with negative component at each outgoing edge, hence there is no ambiguity. Finally, in **b)** *the* packet with maximum remaining length at each outgoing edge is unique, since all these packets are moving along their last direction (their negative component is already finished, otherwise they would be in **a)**) and each node is the destination of at most one packet.

Using this algorithm, at every step all packets with maximum remaining distance move, and hence at every step the maximum remaining distance over all packets decreases by one. Thus, the total running time is at most ℓ_{max}, meeting the lower bound. More details can be found in [12]. In the case of permutation routing, we have proved that the number of steps i at each node is at most ℓ_{max}, but written in this way the algorithm can be applied in a more general routing scenario. In conclusion, the main result can be summarized in

Theorem 2 *Algorithm $\mathcal{A}$ is an optimal permutation routing algorithm for full-duplex hexagonal networks.*

Besides minimizing the number of steps, a routing algorithm must also be easy to implement; namely, the routing at each step should be determined efficiently. A routing algorithm is called *oblivious* [5, 6] if the path of v for each node v depends only in v and its destination, although the waiting time at an intermediate node may depend on other paths. On the other hand, a *translation invariant* oblivious algorithm is completely determined by paths from the origin.

The obliviousness of $\mathcal{A}$ is straightforward since the routing for each packet depends only on the origin and destination nodes. Finally, it is clear that to route a packet only the difference $D - S$ between the source and destination node is necessary, and thus we have proved the invariance.

Corollary 1 *Algorithm $\mathcal{A}$ is oblivious, translation invariant and an optimal permutation routing algorithm for full-duplex hexagonal mesh networks.*

5 Algorithm for Square Grid

Many communication networks are represented by graphs satisfying the following property: for any pair of nodes u and v, the edges of a shortest path from u to v can be partitioned into k disjoint classes according to a well-defined criterium. For instance, we have seen in Section 4 that on a hexagonal network the edges of a shortest path can be partitioned into *positive* and *negative* ones. Similarly, on a k-circulant graph the edges can be partitioned into k classes according to their length.

In graphs that satisfy this property there exists a natural routing algorithm: route all packets along one class of edges after another. We have proved that for hexagonal networks this algorithm turns out to be optimal. Optimality for 2-circulant graphs is proved using a static approach in [5], and recently using a dynamic distributed algorithm in [11]. In [5] the authors introduce the notion of *big-foot* algorithms since their algorithm routes packets first along *long hops* and then along *short hops* in a 2-circulant graph.

On the square grid, the *big-foot* idea works as it is natural. I.e., an optimal algorithm consists of 2 phases, moving each packet first horizontally and then vertically. These two phases assure that a packet may wait, possibly, during the second direction of its path. Since all destinations are distinct, there is no additional delay, and thus the same proof that we have provided for triangular grid in Section 4.3 guarantees the optimality for square grid. Details are straightforward and are omitted due to space limitation. Summarizing, it can be proved that

Theorem 3 *There is a translation invariant oblivious optimal permutation routing algorithm for full-duplex networks that are convex subgraphs of the infinite square grid.*

6 Algorithm for Hexagonal Grid

The first observation about this topology is that, even in the full-duplex case, no algorithm can finish in ℓ_{max} steps, as we can see with a counterexample in Figure 3. Small letters (a and b) denote source nodes, while capital letters (A and B) denote destination nodes. Both packets have paths with length 4, but it is not possible to route both packets in less than 5 steps. Examples with larger delays exist. Hence, no algorithm can guarantee all communications to be finished in ℓ_{max} communication steps.

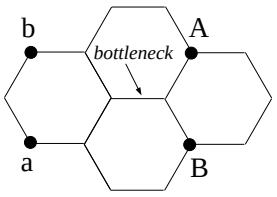

Figure 3: Counterexample showing that no algorithm can achieve ℓ_{max} steps in the hexagonal grid

Theorem 4 *There is a translation invariant oblivious permutation routing algorithm for full-duplex networks that are convex subgraphs of the infinite hexagonal grid which needs at most $2\ell_{max}$ steps.*

Since ℓ_{max} is obviously a lower bound, the algorithm provides a 2 factor approximation.

Sketch of proof. The nodes of the hexagonal grid can be assigned three coordinates in the same way as in the triangular grid (the difference is that one third of nodes and one half of edges are missing). Shortest paths can be read from the relative address $D-S$, and the particular shortest path used is chosen as follows. First, define

6 directions which may be regarded as directions to neighbors of distance two, i.e. $N = (1, -1, 0), NE = (1, 0, -1), SE = (0, 1, -1), S = (-1, 1, 0), SW = (-1, 0, 1)$, and $NW = (0, -1, 1)$.

It can be shown that any shortest path can be realized by a sequence of double moves in only two among the above directions plus possibly one single move. More precisely, for any relative address, one can find a path of one of the forms: $S + SW, WS + NW, NW + N, N + NE, NE + SE$, or $SE + S$.

In an auxiliary graph which realizes these double moves, one can use the ideas that work for triangular grid. (More precisely, the auxiliary graph is formally the square of the original graph.) For realization of the routing in the original graph, one has to resolve the possible conflicts. For example, SE and NE share one edge, and can not be applied simultaneously. However, a simple trick that evenly switches the priorities at most doubles the number of communication steps needed. The details are omitted due to space limitations and will be given elsewhere.

References

[1] J. Cogolludo and S. Rajasekaran. Permutation Routing on Reconfigurable Meshes. *Algorithmica*, 31:44–57, 2001.

[2] A. Datta. A Fault-Tolerant Protocol for Energy-Efficient Permutation Routing in Wireless Networks. *IEEE Transactions on Computers*, 54(11):1409–1421, November 2005.

[3] T. Dobravec, J. Žerovnik, and B. Robič. Permutation routing in double-loop networks: design and empirical evaluation. *Journal of Systems Architecture*, 48:387–402, 2003.

[4] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53:79–134, 1994.

[5] F. Hwang, T. Lin, and R. Jan. A Permutation Routing Algorithm for Double Loop Network. *Parallel Processing Letters*, 7(3):259–265, 1997.

[6] F. Hwang, Y. Yao, and B. Dasgupta. Some permutation routing algorithms for low-dimensional hypercubes. *Theoretical Computer Science*, 270:111–124, 2002.

[7] G. E. Jan and M.-B. Lin. Concentration, load balancing, partial permutation routing, and superconcentration on cube-connected cycles parallel computers. *J. Parallel Distrib. Comput.*, 65:1471–1482, 2005.

[8] D. Karimou and J. F. Myoupo. An Application of an Initialization Protocol to Permutation Routing in a Single-Hop Mobile Ad Hoc Networks. *The Journal of Supercomputing*, 31:215–226, 2005.

[9] W. Liang and X. Shen. Permutation Routing in All-Optical Product Networks. *IEEE Transactions on Circuits and Systems*, 49(4):533–538, 2002.

[10] F. G. Nocetti, I. Stojmenović, and J. Zhang. Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):963–971, 2002.

[11] B. Robič and J. Žerovnik. Minimum 2-terminal routing in 2-jump circulant graphs. *Computers and Artificial Intelligence*, 19(1):37–46, 2000.

[12] I. Sau and J. Žerovnik. An Optimal Permutation Routing Algorithm for Full-Duplex Hexagonal Mesh Networks. *Preprint series, University of Ljubljana, Institute of Mathematics, Physics and Mechanics*, 44(1017), 2006. ISSN 1318–4865.

[13] I. Stojmenović. Honeycomb Networks: Topological Properties and Communication Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 8(10):1036–1042, 1997.

[14] R. Trobec. Two-dimensional regular d-meshes. *Parallel Computing*, 26:1945–1953, 2000.