Parallel Processing Letters,
© World Scientific Publishing Company

PURE GREEDY HOT-POTATO ROUTING IN THE 2-D MESH WITH RANDOM DESTINATIONS

PAUL SPIRAKIS
Computer Technology Institute, Patras University
Patras, PO BOX 1122, 26110, Greece
e-mail : spirakis@cti.gr

and

VASSILIS TRIANTAFILLOU
Computer Technology Institute, Patras University
Patras, PO BOX 1122, 26110, Greece
e-mail : triantaf@cti.gr

Received
Revised
Communicated by

ABSTRACT
We analyze here a *pure* greedy hot-potato routing strategy on a two-dimensional mesh of n^2 nodes. We specifically study the case of n^2 packets, originating one per node, to be delivered at random uniform destinations. Each packet attempts to follow the shortest path leading first to the destination row/column (whichever is closest) and then to the actual destination node. A deflection policy is adopted to solve conflicts. We prove that *all* packets are delivered to the destinations in *average time $O(nlogn)$*. The average is taken over *all* possible destination functions. No average case analysis of pure greedy hot-potato routing was known up to now.

Keywords: Hot-potato routing, Two-Dimensional Mesh, Greedy Algorithms, Random Destinations

1. Introduction

In this work we propose a pure greedy algorithm for packet routing in a synchronous 2-dimensional mesh of processors in which at most one packet can traverse any directed link in each time step. We consider a class of algorithms known as *hot-potato* (or *deflection* routing algorithms (see e.g. [1], [2]) The important characteristic of these algorithms is that they use no buffer space for storing delayed packets. Each packet, unless it has already arrived to its destination, must leave the node (processor) at the step following its arrival. This may cause some packets to be *deflected* away from their preferred direction.

Variants of the hot potato routing are used by parallel machines such as the HEP multiprocessor and the Connection Machine and by high speed communication networks [3]. In particular, hot potato routing is very important in fine-grained massively-parallel computers, such as the Caltech Mosaic C [4]. For such machines the addition of even a small sized storage buffer at each processor will cause a substantial increase in the cost of the machine. Deflection-type routing is highly desirable in *optical* networks (see e.g. [2], [5]) or other very fast networks : this is so because any intermediate storage must take an electronic form and then be converted back to optical. This conversion is *very slow* compared to optical transmission.

Most of the recent work on hot-potato routing focuses on *structured* routing. In structured routing good behaviour is enforced by sending the packets in pre-specified directions. Structured routing was shown in [6] to be asymptotically optimal. However, it penalizes packets which initially are very close to their destination, by sending them to distant regions of the network due to the fixed, pre-specified routes. Also, long unnecessary routes are taken even when the actual number of packets is much smaller than the number of nodes, because the algorithms are not sensitive to the total network load.

In contrast, in *greedy* routings, a packet is bound to use an out-going link in the direction of the destination, whenever such a link is available. Aside fron deflections, greediness helps packets to go to their destinations by following shortest paths.

Another practical aspect of greedy routing is its *simplicity*. The amount of hardware and the cycle time of the routing mechanism (at every processor) depend to large extent on the complexity of the routing algorithm. Structured algorithms are often designed in *phases*, where the algorithm changes at each phase. Moreover, the routing choices are usually complex, and might defer for different processors during the same phase. Thus structured algorithms typically rely on complex routing mechanisms. Greedy algorithms, on the other hand, do not consist on phases. All nodes perform the same (simple) routing policy at each step of the algorithm.

Although greediness might cause congestion in certain regions of the network, deflection is hoped to spread the load so that the total routing time is decreased. Simulations of greedy hot potato routing algorithms show their superiority over structured ones. Unfortunately the analysis of greedy hot potato routing algorithms is considerably more difficult than that of the structured ones. There is no bound to the number of links that a packet may traverse before it arrives at its destination. Certain chains of deflections may eventually result back in the original configuration, thus raising the question whether the algorithm ever terminates. Such infinite loops are called *livelock*.

2. Related work and our results

The first greedy hot-potato algorithm was proposed by Baran [7]. Borodin and Hopcroft in [8] suggested a greedy hot-potato algorithm for the hypercube. They observed that "experimentally the algorithm appears promising". Since then, a lot of *experimental* results on hot-potato greedy routing were published ([9], [2], [10], [3] etc.) all noticing that such routings perform very well in experiments or simulations. Prager [10] showed that the Borodin-Hopcroft algorithm terminates in n steps on the 2^n - nodes hypercube for a special class of permutations. Complete analysis, however, exists merely for *many-to-one* routing problems, in which many packets may be destined to reach the same target. Hajek [11] presented a simple greedy algorithm for the hypercube that runs in $2k+n$ steps, where k is the number of packets in the system. The work of Hajek was simplified and generalized in a work by Brassil and Cruz [12]. They showed a bound of $diam + P + 2(k-1)$ for any network, where $diam$ is the network diameter and P is the length of a walk connecting all destinations in a certain order. Ben-Or, Halevi and Schuster in [13] gave a potential function analysis of greedy routing algorithms on d dimensional arrays. Their result yield, a $8\sqrt{2}\sqrt{kn}$ bound for routing k packets on the two dimensional mesh.

Some recent results concern structured (non-greedy) hot potato *permutation routing.* Feige and Raghavan [1] presented an algorithm for the 2-dimensional mesh that routes most of the routing problems in no more than $2n + O(logn)$ steps. Newman and Schuster [6] presented an algorithm that is based on sorting for permutation routing in the 2-dimensional mash. Their algorithm routes every permutation in $7n+o(n)$ steps, which was improved to $3.5n+o(n)$ by Kaufmann et.al [14]. Bar-Noy et.al. [15] presented a relatively simple algorithm for the 2-dimensional mesh and torus that routes every routing problem in $O(n\sqrt{m})$ steps where m is the maximum number of packets destined to a single column. Kaklamanis, Krizanc and Rao [16] presented an algorithm for permutation routing in the d-dimensional torus that routes most of the permutations within $2n + O(log^2 n)$ steps. Ben-Aroya and Schuster [17] proved a lower bound for deterministic algorithms that "stick" to the surrounding of the destination column or destination row once they get there. Their result yield an $\Omega(n^2)$ lower bound for permutation routing by a large class of algorithms. The algorithms in [1], [16] have some "greedy tendency" included. It is easy to see that they loose it when routing some "worst case" instances, and in those cases their performance degrades.

Ben-Aroya and Schuster, in [17], provided *strongly* greedy hot-potato routing algorithms (in the sense that their algorithms attempt to send packets in good directions whenever possible) and gave a potential function analysis for their algorithms. Their analysis gave the best evacuation time known for delivering all packets to their destinations, when the destinations are *arbitrary* (A batch of k packets with maximal source-to-destination distance d_{max} is delivered in $2(k-1) + d_{max}$. This is, however, still $\Theta(n^2)$ when each node of the mesh is the origin of a packet.

In this paper we analyse the pure greedy hot-potato routing under the assumption of *random* packet destinations. We prove termination (of n^2 packets, originating one per node) in $O(nlogn)$ *expected* time, where the average is taken over *all possible* destination functions. Feige and Raghavan [1] and Kaklamanis et.al. [16] have shown that "greedy-like" routing strategies exhibit an $O(n)$ average time performance in our case, for *most* of the routing problems (destination sets). The advantage of our paper is the use and analysis of a *pure greedy strategy* and that our result holds for *all possible* destination functions.

Our extended simulations [18] indicate an $O(n)$ expected time. Thus, we conjecture that the $O(logn)$ slowdown is possibly due to some slack in the analysis.

3. Our model and definitions

The network we are dealing with is the $n \times n$ 2-dimensional mesh with n columns and n rows and a node at each intersection point. The network edges are unidirectional links between processors, where each pair of neighbouring processors has two edges in opposite directions. The network is synchronous so that moving of packets between neighbouring processors is done at discrete time steps. At each step, a processor may receive up to one packet from each incoming edge and send up to one packet along each outgoing edge. After sending and receiving of packets, some pre-specified, standard operations are performed by the processor on the headers of the incoming packets. Our algorithm uses the ability to read and compare the destination of two packets. There are no buffers at any node, and a packet is never stored at any node, except for its destination node.

We consider the problem of many-to-one packet routing, or simple packet routing, in which each node has a (single) packet to be delivered to another node in the mesh. *The destinations are randomly uniformly chosen.*

Definition 1 *: Given a $n \times n$ mesh containing n^2 packets at n^2 different nodes, a routing problem is a function f which assigns destinations to the packets.*

Definition 2 *:*

- *Let f be a routing problem, A be a hot-potato algorithm, and let p be a packet. Since A is a hot-potato algorithm, p goes forward towards its destination, unless it cannot, in which case it goes backwards (away from it) at each step. We call a packet that goes forward an* **advancing packet***, and a packet that goes backward is called a* **deflected** *packet.*
- *Consider a node S in the mesh. We define drow(p) as the distance between the row of S and the destination row of p and dcol(p) as the distance between the column of S and the destination column of p.*
- *For each packet p contained in S if $drow(p) < dcol(p)$ then we can mark the outgoing column from S leading towards p's destination as the good direction for p. If $drow(p) = 0$ then p is travelling on the destination row (good direction). Otherwise $drow(p) > dcol(p)$, we can mark the outgoing row from S leading towards p's destination as the good direction for p. If $dcol(p) = 0$ then p is on the destination column (good direction). If $drow(p) = dcol(p)$ then both the row and column of S leading towards the destination row or column can be marked as good directions.*

Definition 3 *: A hot potato algorithm is greedy if its decision rule has the property that whenever a packet p is deflected, then the good direction of p is used by an* **advancing** *packet. We say that this advancing packet deflects p.*

Definition 4 *: Our algorithm obeys the following rules :*

- *The algorithm is purely greedy*
- *When two or more packets share the same good direction, the algorithm gives priority to the one that wants to proceed in the same direction it came from.*
- *Deflected packets go to* random *directions*
- *A packet always attempts to follow its good direction, unless it is deflected.*

4. The potential function

The potential function of a packet p at time t is denoted by $\Phi_p(t)$. Every packet has an initial potential equal to $a \cdot n$ ($a > 6$ is a suitable constant). This will be shown to be enough to account for possible deflections until p reaches its row or column and starts to travel towards its destination. From that pointr on, no other packet can stop p. When p reaches its destination, all the remainder potential is lost too. Formally we define $\Phi_p(t)$ as follows:

Definition 5 *:*

- *Initially, for every p, $\Phi_p(0) = \alpha \cdot n$*
- *If at time t the packet p is deflected away from its destination then $\Phi_p(t+1) = \Phi_p(t) + 1$*
- *If p advances towards its destination row or column (the nearest one) then $\Phi_p(t+1) = \Phi_p(t) - 3$*
- *If p reached its destination at time t then $\Phi_p(t)$ is* **forced to zero**

The potential of the mesh at time t, denoted by $\Phi(t)$, is the sum of $\Phi_p(t)$ over all the packets.

We say that a node in the mesh *looses* potential in step t if the sum of the potential over the packets that entered that node at time t is greater than the sum over the same packets at time $t+1$. If the sum at time t is less than the sum at time $t+1$, we say that the node *gains* potential.

Definition 6 *: Let S be a network node. Let A, B, C, D be the situations (events) where S contains one, two, three or four packets.*

Note that when S contains four packets at time t (i.e. situation D) then we may distinguish four subcases:

- *subcase (i)* : All packets go to the right direction, at $t+1$. In this case the potential loss is 12 units.
- *subcase (ii)* : Three packets go to their right direction at $t+1$. In such a case we have a potential loss of 8 units.
- *subcase (iii)* : Two packets go to their right direction. Then the potential loss is 4 units.
- *subcase (iv)* : Only one packet goes to the right direction at time $t+1$. In such a case the potential loss is 0 units.

Let us denote these subcases by D_1, D_2, D_3, D_4.

Definition 7 : *For a node S, let $\Delta P_S(x)$ be the worst case (smallest) potential loss in each case $x \epsilon \{A, B, C, D\}$.*

5. A technical remark and an Assumption

First, note that $\Delta P_S(A) = 3, \Delta P_S(B) = 2, \Delta P_S(C) = 1$ and $\Delta P_S(D) = 0$

In the sequel we assume

Assumption 1 *Packets arriving at a node have random and independent destinations*

Note that such an assumption has been used many times in analyses of network routing. We aim here in an approximate analysis of the progress of the routing scheme.

However, in the artificial scenario assumed by assumption 1, the distribution of all possible destination functions is the same as in the case when packets just select destinations uniformly randomly at the beginning and do not change their choices. Thus, our assumption covers exactly the case of random independent destinations.

Definition 8 *Let $\mathcal{F}_t^S$ be the event "only one packet of node S will move to the correct direction at time $t+1$"(no progress potential)*

Lemma 1 *At any time t, the expected number of the nodes S in situation D satisfying $\mathcal{F}_t^S$ is at most a constant fraction less than 1*

Proof. The lemma trivially holds at time $t = 0$ since no nodes are in situation D. Assume, inductively, that the lemma holds at time t. Consider a node v of type D. In the (worst) case that this node is being surrounded by four nodes of type D, each of them will deflect a packet of a particular destination (out of v) to v (say north) with probability $\frac{1}{3}$. Thus, the probability that v will have all its packets with the same (conflicting) destination at time $t+1$ is at most $4(\frac{1}{3})^3 = \frac{4}{27}$

Let $N_L(t)$ be the set of nodes with at least 1 packet at time t and $|N_L(t)|$ their number. Thus the expected number of nodes S in situation D satisfying $\mathcal{F}_t^S$ is at most $\frac{4}{27} \cdot |N_L(t)|$, by linearity of expectation. □

Corollary 1 *Let $E(\Delta\Phi^S(t+1))$ be the expected value of the potential loss of node S at the beginning of time $t+1$ then*

$$\sum_{S \in N_L(t)} E(\Delta\Phi^S(t)) \geq \lambda \cdot |N_L(t)|$$

where $\lambda > 0$ is a constant, $\lambda \geq \frac{23}{27}$

Proof. Clearly

$$\sum_{S \in N_L(t)} E(\Delta\Phi^S(t)) \geq |N_L(t)| - E(X)$$

where X= number of nodes with at least 1 packet but in situation D and satisfying $\mathcal{F}_t^S$.

But $E(X) \le \frac{4}{27} \cdot |N_L(t)|$ by lemma 1. Thus

$$\sum_{S \in N_L(t)} E(\Delta\Phi^S(t)) \ge \frac{23}{27} \cdot |N_L(t)|$$

$\Box$.

Consider now the situation of a particular packet p arriving at node S at time t. Let $s(p)$ be the probability that p will not be affected at S. This probability is minimized when S receives three other links. Let E_1, E_2, E_3, E_4 be the events:

$E_1 =$ all four packets have different destinations
$E_2 =$ p's destination conflicts with one other
$E_3 =$ p's destination conflicts with two others
$E_4 =$ all packets go to the same destination

Let E be the event that p will not be deflected. Then

$$\mathbf{Pr}(E) = 1 \cdot \mathbf{Pr}(E_1) + \frac{1}{2} \cdot \mathbf{Pr}(E_2) + \frac{1}{3} \cdot \mathbf{Pr}(E_3) + \frac{1}{4} \cdot \mathbf{Pr}(E_4)$$

Fix p's destination. Due to our assumption,

$$\mathbf{Pr}(E_1) = \binom{3}{0} \left(1/3 \right)^0 \left(2/3 \right)^3 = \frac{8}{27}$$

(because each other packet independently selects p's destination with probability $\frac{1}{3}$ and thus we count the number of successes in Bernoulli trials). Similarly

$$\mathbf{Pr}(E_2) = \binom{3}{1} \left(1/3 \right) \left(2/3 \right)^2 = \frac{4}{9} = \frac{12}{27}$$

$$\mathbf{Pr}(E_3) = \binom{3}{2} \left(1/3 \right)^2 \left(2/3 \right) = \frac{2}{9}$$

and

$$\mathbf{Pr}(E_4) = \binom{3}{3} \left(1/3 \right)^3 = \frac{1}{27}$$

Thus

$$s(p) \ge \frac{8}{27} + \frac{1}{2} \cdot \frac{12}{27} + \frac{1}{3} \cdot \frac{2}{9} + \frac{1}{4} \cdot \frac{1}{27} > \frac{16}{27} > \frac{1}{2}$$

Thus we have shown

Lemma 2 *For each packet p and time t, the probability that the packet will not be deflected upon its arrival at the current node is greater than* $\frac{16}{27} > \frac{1}{2}$

Corollary 2 *At each step, the expected number of packets that will not be deflected is strictly more than half of them.*

Corollary 3 *The total expected potential of the grid is reduced at each step.*

Proof: At least half the packets in motion are expected not to be deflected. Each reduces its potential by 3. In the worst, the rest (less than half) increase each its potential by 1. Thus, the expected loss of the total potential is always positive. □

From Lemma 2, we also have

Corollary 4 *The expected potential of any individual packet p is reduced at each step, hence it is always $O(n)$.*

Proof: For each packet p

$$\Phi_p(t+1) = (\Phi_p(t)+1)\cdot(1-s(p)) + (\Phi_p(t)-3)\cdot s(p)$$

and by getting expectations,

$$E(\Phi_p(t+1)) \leq E(\Phi p(t) - 2)$$

□.

Remark 1 *Also, note that, due to Lemma 2, each packet p is expected to succeed not to be deflected in at least 16 out of every 27 node visits. This means that, by choosing the constant α to be e.g. greater than 9, the expected potential of p will never become zero before p arrives to its destination.*

6. The progress of the routing scheme

Note that $\Phi(0) = \Theta(n^3)$ is the initial total potential

Definition 9 *Let T be the total number of routing steps*

Decompose T as

$$T = \sum_{i=0}^{2logn} T_i$$

where T_i = the number of steps (phase i) during which the nodes with at least one packet are at least $n^2/2^{i+1}$ and at most $n^2/2^i$. Clearly,

$$E(T) = \sum_{i=0}^{2logn} E(T_i)$$

by linearity of expectation.

Since each node can have at most four packets per step and since the expected potential of each packet is $O(n)$, due to Corollary 4, then the total expected potential at the beginning of phase i, Φ_i, is at most $O(n^3/2^i)$ by definition of T_i, $\forall i$.

Let $\Delta\Phi_i$ be the potential drop during phase i. Clearly,

$$\Delta\Phi_i = \sum_{t\ in\ phase\ i} \sum_{S \in N_L(t)} \Delta\Phi^S(t)$$

Thus,

$$E(\Delta\Phi_i) = E(\sum_{t\ in\ phase\ i} \sum_{S\in N_L(t)} E(\Delta\Phi^S(t))) \geq E(\sum_{t\ in\ phase\ i} \frac{23}{27} N_L(t))$$

(by Corollary 1)
But, by definition of phase i, $N_L(t) \geq \frac{n^2}{2^{i+1}}$ for all t in phase i. Thus

$$E(\Delta\Phi_i) \geq \frac{n^2}{2^{i+1}} \cdot \frac{23}{27} \cdot E(T_i)$$

But $\Phi_i - \Phi_{i+1} = \Delta\Phi_i$
i.e. $\Delta\Phi_i + \Phi_{i+1} = \Phi_i$ and $E(\Phi_{i+1}) = O(n^3/2^{i+1})$
We then get

$$\frac{n^2}{2^{i+1}} \cdot \frac{23}{27} \cdot E(T_i) \leq E(\Delta\Phi_i) = E(\Phi i) - E(\Phi i + 1)$$

i.e. $\frac{n^2}{2^{i+1}} \cdot \frac{23}{27} \cdot E(T_i) \leq O(\frac{n^3}{2^{i+1}})$, because the total potential is always nonnegative and becomes zero only when all packets reach their destination, due to our previous Remark 1. Thus $E(T_i) = O(n)\ \forall i$. Hence

$$E(T) = \sum_{i=0}^{2logn} E(T_i) = O(nlogn)$$

Thus we have shown

Theorem 1 *The average routing time of the greedy hot-potato routing to deliver all packets to random destinations is $O(nlogn)$*
□

Remark 2 *Note that the arguments used in the analysis do not depend much on the specific rules of the routing protocol given in Definition 4. Thus, our analysis applies to any protocol where at every step each packet attempts to follow some good direction that leads it closer to its destination.*

7. Future work

We conjecture here that our result of $\Theta(nlogn)$ expected termination time for n^2 packets with random destinations is not optimal. Our experiments [18] in a 2D-mesh of a 512-node Parsytec machine indicate a $\Theta(n)$ hot-potato greedy termination time for n^2 packets.

Acknowledgments

This work was partially supported by the EU ESPRIT Basic Research Projects ALCOM IT (contract no. 20244) and GEPPCOM (contract no. 9072).

References

[1] U. Feige and P.Raghavan, Exact analysis of hot-potato routing, In *IEEE Symp. on foundations of Computer Science*, (November 1992).

[2] A.G. Greenberg and J. Goodman, Sharp approximate models of deflection routing in mesh networks, *to appear in IEEE Trans. Communications*, (1992).

[3] N.F. Maxemchuk, Comparison of deflection and store and forward techniques in the Manhattan street and shuffle exchange networks, In *IEEE INFOCOM*, (1989) 800–809.

[4] C.L. Seitz, N. Boden, J. Seizovic, and W. Su, The design of the Caltech Mosaic C multiprocessor, In *Proc. Symp. on Integrated Systems*, (1993) 1–22

[5] Z. Zhang and A.S. Acampora, Performance analysis of multihop lightwave networks with hot potato routing and distance age priorities, In *Proc. IEEE INFOCOM*, (1991) 1012–1021.

[6] I. Newman and A. Schuster, Hot-potato algorithms for permutation routing, Technical Report PCL Report #9201, CS dept, Technion, November 1992.

[7] P. Baran, On distributed communication networks, *IEEE Transactions on Communications*, (1964) 1–9.

[8] A. Borodin and J.E. Hopcroft, Routing, merging, and sorting on parallel models of computation, *Journal of Computer System Sciences*, , **30** (1985) 130–145.

[9] A.S. Acampora and S.I.A. Shah, Multihop lightwave networks: a comparison of store-and-forward and hot-potato routing, In *IEEE INFOCOM*, (1991) 10–19

[10] R. Prager, An algorithm for routing in hypercube networks, PhD thesis, University of Toronto, 1986.

[11] B. Hajek, Bounds on evacuation time for deflection routing, *Distributed Computing*, **5** (1991) 1–6.

[12] J.T. Brassil and R.L. Cruz, Bounds on maximum delay in networks with deflection routing, In *29th annual Allerton conf. on Communication, Control and Computing*, (1991) 571–580.

[13] A. Ben-Or, S. Halevi and A. Schuster, On Greedy Hot-Potato Routing, Manuscript, 1993.

[14] M. Kaufmann, H. Layer, and H. Schroeder, Fast deterministic hot-potato routing on processor arrays, In *ISAAC*, 1994

[15] A. Bar-Noy, P. Raghavan, B. Schieber, H. Tamaki, Fast deflection routing for packets and worms, In *Proc. 12th ACM Symp. on Principles Of Distributed Computing*, 1993

[16] C. Kaklamanis, D. Krizanc, and S. Rao, Hot-potato routing on processor arrays, In *Symp. of Parallel Algorithms and Architectures* , 1993

[17] I. Ben-Aroya and Assaf Schuster, Greedy Hot-Potato Routing on the Two-Dimensional Mesh, In *Proc. 2nd European Symp. on Algoritms (ESA 94)*, LNCS

[18] S. Karaivazoglou, P. Spirakis and V. Triantafillou, Wormhole versus deflection routing : A case study on the mesh, *In COCOON '96 Conference*