A Note on Deflection Routing on Undirected Graphs

Technical Report Number 493

November 1994

Antonios Symvonis
A Note on Deflection Routing on Undirected Graphs

Antonios Symvonis
Basser Department of Computer Science
University of Sydney
Sydney, N.S.W. 2006
Australia
symvonis@cs.usyd.edu.au

Technical Report 493
November 1994

Abstract

We provide an algorithm that routes a load of \(k\) packets on an undirected graph \(G\) within \(2d_{am_G} + 2(k - 1)\) routing steps where \(diam_G\) is the diameter of graph \(G\). For a \(d\)-dimensional mesh \(M\) the required number of routing steps is at most \(diam_M + (d - 1) + 2(k - 1)\) where \(diam_M\) is the diameter of mesh \(M\). The algorithm satisfies the “one-pass of links” property.
1 Introduction

In this note, we study the routing of messages in undirected graphs. Message routing has been abstracted in several ways. In packet routing it is assumed that a message can be transmitted between two adjacent processors (vertices of the undirected graph) in a single step as a packet. We examine the routing model known as deflection (or hot-potato) routing in which packets continuously move between processors from the time they are injected into the network until the time they are consumed at their destination.

The advantage of deflection routing is obvious. No queueing area is required at the processors. However, the fact that packets always move implies that at any step each processor must transmit the packets it received during the previous step (unless they were destined for it). As a result, several packets might be derouted away from their destination. This makes the analysis extremely difficult.

The work of Feige and Raghavan [4] which provided analysis for deflection routing algorithms for the torus and the hypercube, renewed the interest in deflection routing. As a result, several papers appeared with deflection routing as their main theme. Bar-Noy et al [1] gave a nearly optimal deterministic algorithm for routing permutations on the \( n \times n \) mesh and torus. Based on sorting algorithms, Newman and Schuster [8] derived asymptotically optimal algorithms for routing permutations on the mesh and the torus networks and a near optimal algorithm for the hypercube. Kaufmann et al [7] fine tuned their results for the case of meshes and managed to reduce the constant hidden in the asymptotic analysis.

Ben-Aroya and Schuster [2] considered the general situation where a mesh is loaded with \( k \) packets that have to be routed to their destinations in a hot-potato manner. Through clever analysis of their greedy algorithm they showed that routing will terminate within \( dist + 2(k - 1) \) steps where \( dist \) is the initial maximum distance a packet has to travel. Independently, Borodin et al [3] formalized the notion of the deflection sequence, a nice way to charge each deflection of an individual packets to distinct packets traveling on the network. By using their method, they show that routing \( k \) packets in a hot-potato manner can be completed within \( dist + 2(k - 1) \) steps for trees, butterflies and multidimensional meshes, where \( dist \) is the initial maximum distance a packet has to travel. For two dimensional meshes, even though their result is the same with that of Ben-Aroya and Schuster [2], their algorithm is considered to be superior since it is has better performance for the significant case of permutation routing\(^1\).

In their paper, Borodin et al [3] mention several open problems. Among them, i) the problem (attributed to Hajek [6]) of deriving a deflection routing algorithm for an arbitrary undirected graph which routes any routing pattern consisting of \( k \) packets within \( dist + 2(k - 1) \) steps where \( dist \) is the initial maximum distance a packet has to travel, and ii) whether the “one-pass of links” property (as defined by Feige and Raghavan [4]) is a substantial restriction to fast deflection routing.

In this research note we build on the research of Borodin et al [3] and we describe a simple algorithm which satisfies the “one-pass of links” property and routes any \( k \) packet routing pattern on any undirected graph \( G \) within \( 2diam_G + 2(k - 1) \) routing steps where \( diam_G \) is the diameter of graph \( G \). Besides of coming very close to the desired \( diam_G + 2(k - 1) \) routing performance, our result also shows that the “one-pass of links” property is not a major restriction for routing on undirected graphs (note that there exist routing patterns on trees which require \( diam_G + 2(k - 1) \) steps for their routing). For a \( d \)-dimensional mesh \( M \) the required number of routing steps reduces to \( diam_M + (d - 1) + 2(k - 1) \) where \( diam_M \) is the diameter of the mesh \( M \).

\(^1\)The Algorithm of Ben-Aroya and Schuster [2] has not been analysed for the special case of permutation routing yet.
2 Deflection Routing on Undirected Graphs

2.1 Preliminaries

We begin with some definitions that are necessary in order to state and analyse our algorithm. We then proceed with the description of our algorithm.

Definition Consider any step of a deflection algorithm and the packets assigned to outgoing links of any node at that step. The deflection algorithm is called greedy if there is not a pair of packets for which a swap of the assigned links will result to a reduction to the number of deflections at that node at that

Definition A deflection routing algorithm is said to satisfy the one-pass of links property if the order in which packets are assigned to outgoing links at a given node is based on a fixed order of the incoming links through which the packets arrived at the node.

Borodin et al [3] defined the notions of the deflection sequence and the deflection path as follows: Consider a deflection of a packet \( p \) at time \( t_1 \) and let \( p_1 \) which caused the deflection. Follow packet \( p_1 \) untill time \( t_2 \) where it reaches its destination or it meets with packet \( p_2 \). In the latter case, follow packet \( p_2 \) untill time \( t_3 \) where it reaches its destination or it meets with packet \( p_3 \), and so on. We continue in this manner untill we follow a packet \( p_i \) which reaches its destination at time \( t_{i+1} \). The sequence of packets \( p_1, p_2, \ldots, p_i \) is defined to be the deflection sequence corresponding to the deflection of packet \( p \) at time \( t_1 \). The path (starting from the deflection node and ending to the destination of \( p_i \)) which is defined by the deflection sequence is said to be the deflection path corresponding to the deflection of packet \( p \) at time \( t_1 \).

Lemma 1 (Borodin, Rabani, Schieber [3]) Suppose that for any deflection of packet \( p \) from node \( v \) to node \( u \) the shortest path from node \( u \) to the destination of \( p \) (the last packet in the deflection sequence) is at least as long as the deflection path. Then, \( p_1 \) cannot be the last packet in any other deflection sequence of packet \( p \). Consequently we can “charge” the deflection to packet \( p_1 \).

2.2 The algorithm

We will use a spanning tree of the graph to perform the routing. We assume that each node has a global view of the spanning tree, i.e., it is able to determine which of its adjacent edges are part of the spanning tree and whether these edges advance a given packet towards its destination (if routing is restricted to spanning tree edges only). These requirements are reasonable, since the spanning tree has to be computed only once for a given graph.

Let \( G = (V, E) \) be the graph in which the routing is supposed to take place and let \( T = (V, E') \) be a spanning tree of \( G \). By \( \text{diam}_G \) and \( \text{diam}_T \) we denote the diameter of \( G \) and \( T \), respectively. Let \( v \) be a node of \( G \). We denote by \( d_G(v) \) the degree of node \( v \) in \( G \) and by \( d_T(v) \) the degree of node \( v \) in the spanning tree \( T \) of \( G \). By \( I_G(v) \) we denote the edges of \( G \) which are incident to \( v \) while by \( I_T(v) \) we denote the edges of the spanning tree \( T \) which are incident to \( v \).

We describe the routing algorithm from the point of view of a node \( v \). We specify how packets which were received during the previous step are assigned to outgoing links (and transmitted) in the current step. For clarity in the presentation we will talk about incoming and outgoing links. This does not

\footnote{Several times in this research note we will use the phrase “packets are assigned to links in a greedy manner” to indicate that routing assignments are done by a greedy algorithm.}
Algorithm *Deflection-Route*(v)

1. [Initialization, at time \( t = 0 \)]
   - If \( \lambda(v) \leq d_T(v) \) assign all \( \lambda(v) \) packets to outgoing spanning tree edges in a greedy manner.
   - If \( \lambda(v) > d_T(v) \) assign \( d_T(v) \) packets to outgoing spanning tree edges in a greedy manner and assign the remaining \( \lambda(v) - d_T(v) \) packets to edges in \( I_G(v) - I_T(v) \).
   - Call all packets assigned to a spanning tree edge primary packets and all those assigned to non-spanning tree edges secondary packets.

2. [At time \( t > 0 \)]
   - Assign the primary packets received in the previous step in a greedy manner to outgoing spanning tree edges. (Packets which arrived in the previous step and are destined for \( v \) are “consumed”.)
   - If \( t = 2k + 1 \), \( k \geq 0 \), send each secondary packets received in the previous step back to the node from which it arrived.
   - If \( t = 2k \), \( k > 0 \), and \( \lambda_{t-1}(v) \) primary packets were received in the previous step, assign in a greedy manner \( d_T(v) - \lambda_{t-1}(v) \) to the available spanning tree edges (and thus, promote them to primary packets) and send each of the remaining secondary packets back to the node from which it arrived.

**Theorem 1** Assume a graph \( G \), and an initial load of \( k \) packets that have to be routed. Moreover assume that the number of packets which are initially in any node of \( G \) is at most as large as the degree of that node. Then, if routing is performed by algorithm Deflection-Route, all packets will reach their destination within \( \text{diam}_T + 2(k - 1) \) steps, where, \( \text{diam}_T \) is the diameter of the spanning tree which algorithm Deflection-Route uses for the routing.

**Proof** We prove the theorem by constructing deflection paths for the deflections of each packet \( p \) that satisfy the requirements of Lemma 1, i.e., if packet \( p \) is deflected from node \( v \) to node \( u \), then the shortest path from node \( u \) to the destination of \( p \) (the last packet in the deflection sequence) is at least as long as the deflection path.

We first consider the deflections of primary packets. Let \( p \) be such a packet that is deflected at node \( v \) at time \( t_1 \). Since the assignment of the links was greedy, there must exist a packet \( p_1 \) which advances from node \( v \) at time \( t_1 \). We start the deflection sequence with \( p_1 \) which we follow until it reaches its destination or until it is deflected at time \( t_2 \). Since it was deflected, there exists a packet \( p_2 \) which advances from the node where the deflection took place at time \( t_2 \). We continue the deflection sequence with packet \( p_2 \) which we follow until it reaches its destination or until it is deflected, and so on. The deflection sequence is completed when the packet \( p_1 \) which we follow reaches its destination. The deflection path starts at node \( v \) and ends at the destination of \( p_1 \). Moreover, it never uses the same edge twice, i.e., it is a shortest path between its endpoint in the spanning tree in which the routing takes place. So, the shortest path from \( u \) to the destination of \( p_1 \) is 1 edge longer than the deflection path.

Consider now a packet \( p \) which is a secondary packet. (Note that packet \( p \) will later become a primary packet. The deflections of \( p \) as a primary packet have already been considered). At this point we should make clear what we consider to constitute a deflection of a secondary packet. We treat as a deflection of a secondary packet its failure to be promoted to a primary packet. Since secondary packets can be promoted only at their node of origin (during even time steps), all deflections of secondary packets occur at their origin nodes. Our decision to treat as deflections of secondary packets their failures to be promoted to primary ones, is consistent with the fact that we consider the cost of each deflection to be 2 steps. Now
consider a packet $p$ that failed to be promoted at its origin node $v$ at an even step $t$ of our algorithm and as a result it was sent through an edge of $I_G(v) - I_{T}(v)$ to a node $u$. Since in node $v$ at the time of the deflection all spanning tree edges were assigned to packets in a greedy manner, there must exist a packet $p_l$ which advances towards its destination. We start the deflection sequence from $p_l$ and we built it in exactly the same way as in the case of primary packets. Again the deflection path is a shortest path between $v$ and the destination of $p_l$, the last packet in the sequence. The shortest path which our algorithm allows packet $p$ to take in its trip from node $u$ to the destination of $p_l$ has to pass through node $v$. Thus, it is $1$ edge longer than the deflection path.

Since we have showed that for each deflection of any distinct packet $p$ we are able to construct a deflection path that satisfies the requirements of Lemma 1, we can charge each of the deflections of $p$ to a distinct packet. This implies that $p$ will reach its destination through spanning tree edges after at most $dist(p) + 2(k - 1)$ steps, where $dist(p)$ is the initial distance that $p$ has to travel to its destination (through) the spanning tree. The fact that $dist(p) \leq diam_T$ implies the theorem. 

Note that the algorithm satisfies the “one-pass of links” property. Not all algorithm which satisfy the “one-pass of links” property can be greedy. However, this is possible in the case of routing on trees since each packets can advance towards its destination through only one tree edge.

Also note that several variations of the algorithm achieve the same result. For example, it is not necessary to promote secondary packets to primary ones only in their origin node. However, this will slightly complicate the analysis since we will have to deal with deflections which cost one routing step instead of two. We have chosen to present the algorithm in the way we did for clarity reasons and in order to make the best use of the charging scheme of Borodin et al[3] in its original form (Lemma 1).

**Lemma 2** For any connected graph $G$ of diameter $diam_G$ there exists a spanning tree $T$ of $G$ with diameter $diam_T \leq 2diam_G$.

**Proof** The spanning tree obtained by a breadth first search of graph $G$ satisfies the property. 

Note that all spanning trees of a circuit (a connected simple graph in which each node is of degree 2) of $n$ nodes have diameter $n - 1$. The diameter of the circuit is $\lfloor \frac{n}{2} \rfloor$. Finding a spanning tree of minimum diameter is NP-complete [5].

**Corollary 1** Assume a graph $G$ of diameter $diam_G$, and an initial load of $k$ packets that have to be routed. Moreover assume that the number of packets which are initially in any node of $G$ is at most as large as the degree of that node. Then, algorithm Deflection-Route routes all packets to their destinations within $2diam_G + 2(k - 1)$ steps.

**Proof** Follows from Theorem 1 and Lemma 2. 

**Lemma 3** Consider any two dimensional $a \times b$ mesh $M$, $a \leq b$. Then, if either $a$ or $b$ is odd, then there exists a spanning tree $T$ of $M$ of diameter $diam_T = diam_M$. In the case where both $a$ and $b$ are even, there exists a spanning tree $T$ of $M$ of diameter $diam_T = diam_M + 1$.

**Proof** We consider 2 cases based on the parities of $a$ and $b$. For each case the mesh is partitioned into 4 regions (north, east, south, west) by a spanning tree “backbone”. Nodes to the east and west regions are connected to the backbone through horizontal connections while nodes in the north and south regions are connected to the backbone through vertical connections.

**Case 1.** $a$ is odd or $b$ is odd. The spanning tree backbone for the case where both $a$ and $b$ are odd is shown in Figure 1.a. The highlighted node has coordinates $(\lfloor \frac{a}{2} \rfloor, \lfloor \frac{b}{2} \rfloor)$. The backbone for the case where
Figure 1: Spanning tree backbones for 2- dimensional meshes based on the parities of the sidelengths.

$a$ is even is shown in Figure 1.b. The highlighted nodes have coordinates \((\frac{n}{2}, \lfloor \frac{m}{2} \rfloor)\) and \((\frac{n}{2} + 1, \lfloor \frac{m}{2} \rfloor)\). The backbone for the case where $b$ is even is shown in Figure 1.c. The highlighted nodes have coordinates \((\lceil \frac{n}{2} \rceil, \frac{b}{2})\) and \((\lceil \frac{n}{2} \rceil, \frac{b}{2} + 1)\). It is easy to verify that the path between any pair of vertices in the constructed spanning tree is of length at most as large as the diameter of the mesh.

**Case 2. Both $a$ and $b$ are even.** The spanning tree backbone for this case is shown in Figure 1.d. The highlighted nodes have coordinates \((\frac{n}{2}, \frac{b}{2}), (\frac{n}{2}, \frac{b}{2} + 1), (\frac{n}{2} + 1, \frac{b}{2})\), and \((\frac{n}{2} + 1, \frac{b}{2} + 1)\). It can be easily verified that the only nodes which are at distance $\text{diam}_M + 1$ of each other (in the spanning tree) are the south-east and the south-west corners of the mesh. The distance between any other pair of nodes is smaller that $\text{diam}_M$.

**Corollary 2** Assume a two dimensional mesh $M$ of diameter $\text{diam}_M$, and an initial load of $k$ packets that have to be routed. Moreover assume that the number of packets which are initially in any node of $G$ is at most as large as the degree of that node. Then, algorithm Deflection-Route routes all packets to their destinations within $\text{diam}_M + 1 + 2(k - 1)$ steps.

**Proof** Follows from Theorem 1 and Lemma 3.

**Lemma 4** Consider a $d$-dimensional $M$. Then, there exists a spanning tree $T$ of $M$ of diameter $\text{diam}_T \leq \text{diam}_M + (d - 1)$.

**Proof** Follows from a generalization of Lemma 3 to more than 2 dimensions. Note that the equality holds when all sidelengths of the mesh are even.

**Corollary 3** Assume a $d$-dimensional mesh $M$ of diameter $\text{diam}_M$, and an initial load of $k$ packets that have to be routed. Moreover assume that the number of packets which are initially in any node of $G$ is at most as large as the degree of that node. Then, algorithm Deflection-Route routes all packets to their destinations within $\text{diam}_M + (d - 1) + 2(k - 1)$ steps.
3 Discussion

In this research note we showed that we can route on an undirected graph $G$ any routing pattern which involves $k$ packets within $2 \text{diam}_G + 2(k-1)$ with a deflection routing algorithm which satisfies the “one-pass of links” property. For the special case of a $d$-dimensional mesh $M$ routing is completed within $\text{diam}_M + (d-1) + 2(k-1)$ steps.

The behaviour of our algorithm is very poor for the case of routing permutations. This can be easily seen for the case of routing on $n \times n$ meshes. It is quite easy to construct permutations in which all packets have to cross a specific edge and thus requiring $\Omega(n^2)$ routing steps.

References


