Geometric Approach for Optimal Routing on Mesh with Buses

Yosi Ben-Asher
Ilan Newman
Dept. of Mathematics and CS, Haifa University

Abstract

The architecture of 'mesh of buses' is an important model in parallel computing. Its main advantage is that the additional broadcast capability can be used to overcome the main disadvantage of the mesh, namely its relatively large diameter. We show that the addition of buses indeed accelerates routing times. Furthermore, unlike in the 'store and forward' model, the routing time becomes proportional to the network load, resulting in decreasing in routing time for a smaller number of packets.

We consider 1-1 routing of m packets in a d-dimensional mesh with n^d processors and $d \cdot n^{d-1}$ buses (one per row and column). The two standard models of accessing the buses are considered and compared: CREW, in which only one processor may transmit at any given time on a given bus, and the CRCW model in which several processors may attempt to transmit at the same time (getting a noise signal as a result). We design a routing algorithm that routes m packets in the CREW model in $O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$ steps. This result holds for $m \leq n^{\frac{2d}{3}}$ for $d \geq 3$ and unconditionally for $d = 2$. A matching lower bound is also proved. In the CRCW case we show an algorithm of $O(m^{\frac{1}{d}} \log n)$ and a lower bound of $\Omega(m^{\frac{1}{d}})$. It is shown that the difference between the models is essentially due to the improved capability of estimating threshold functions in the CRCW case.

1 Introduction

Two basic forms of communication are used in parallel architectures: messages through point-to-point connections and broadcasting through buses or Ethernet media. The possibility of combining both forms of communication is attractive for mesh networks, where it can be used to overcome the mesh's main disadvantage, namely its large diameter. In the d-dimensional mesh with buses, the n^d processors are interconnected by point-to-point links in a rectangular grid of side length n. In addition, each row and column (or lines that are parallel to the axes in a higher dimension) forms a bus which is considered as a broadcast media.

Indeed, experimental machines with mesh of buses have been proposed and implemented. These includes: DAP [4], Orthogonal multiprocessor (OMP) [13], SUPRENUM [12], Aquarius [22], HDFM (3-D grid) [11], Bused hypercube: [9], and Grid of Ethernets (Micronet) [10].

Many papers have considered meshes plus point-to-point communication from the algorithmic point of view [19, 20, 7, 2, 3, 16, 5, 1, 8, 15, 17]. Only a few have considered the problem of routing. Leng and Shende [15], and Rajasekaran [17] showed linear upper and lower bounds for permutation routing on one and two dimensional meshes. These works focused on improving the leading constant by using buses. Results were later improved by Sibeyn et al. in [18] and by Suel in [21]. The work of [17] considers other routing related problems such as $k-k$ and cut-through routing. Most of the other works address mainly selection and semi group computations such as summing or finding the minimum of numbers stored at the different processors. None of these works have considered the sparse case of 1-1 routing (namely, where the number of packets is considerably smaller than the size of the network).

In this paper, we address the fundamental problem of 1-1 routing. An instance of the 1-1 routing problem consists of a set of packets, initially with at most one packet per node, with a set of specified destinations so that no two packets have the same destination. An algorithm for routing is a protocol that runs in each processor and schedules the transmissions of the packets to their destinations. The constraints are that each point-to-point link and each bus can carry at most one message at any step. The problem of 1-1 routing is a generalization of permutation routing that is considered recurrently in the literature (for a comprehensive survey see [14]). As such, it takes into consideration the load of the network as a parameter of reference while considering routing algorithms. On the other hand, it avoids trivial lower bounds that are imposed by the load at individual processors (being the main reason for considering permutation routing too). In addition, 1-1 routing is a building block for many other routing schemes and algorithms (e.g $k-k$ routing and self simulation).

In the case of point-to-point communication a natural lower bound on routing is the network diameter. Moreover, many common networks have routing algorithms whose complexity is indeed linear in the diameter (see [14]). The motivation of this work is to exploit the addition of buses to accelerate routing and to overcome the lower bound of the diameter. In general, one should expect that routing time would be faster when the number of packets is small. As pointed out above, this is not true for routing on meshes because of the diameter lower bound. The addition of buses overcomes this barrier and allows for a more comprehensive optimality; the routing time decreases as the number of packets in the system and decreases. As far as we know, this is the first time that the load of the network (the number of packets) is taken into account and exploited to drive faster routing algorithms.

Incorporating buses into a network requires a mechanism for conflict resolution. There are two different popular models for this: the CREW model [3], where at most one processor can broadcast on a bus at any step, and the stronger CRCW model, in which a special signal is heard on simultaneous broadcasts. For both models we give essentially tight upper bounds for the 1-1 routing problem.

Let n be the side length of a d-dimensional mesh and m the number of packets to be routed. We show that:

- Routing of up to m packets on a d dimensional mesh with CREW buses can be done in $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps for $d = 2$. A matching lower bound is also proven. For $d \geq 3$ we show an upper bound of $O(\lceil (m/n^{\frac{2d}{3}}) \rceil (m^{\frac{1}{d}} + n^{\frac{1}{d+1}}))$.

- On the other hand, we show that for CRCW, using only buses already give near optimal results namely, 1-1 routing of m packets on the two dimensional mesh can be done in $O(m^{\frac{1}{2}} \log n)$ steps using only buses and in $O(\min(m^{\frac{1}{2}} \log n, m^{\frac{1}{2}} + n^{\frac{1}{2}}))$ steps using buses + point-to-point. For d-dimensional meshes with $d \geq 3$, 1-1 routing can be done in $O(\lceil (m/n^{\frac{2d}{3}}) \rceil m^{\frac{1}{d}} \log n)$ steps using only buses and in $O(\lceil (m/n^{\frac{2d}{3}}) \rceil \min(m^{\frac{1}{d}} \log n, m^{\frac{1}{d}} + n^{\frac{1}{d+1}}))$ when point-to-point communication is used as well.

Both CREW and CRCW show essentially the same behavior: $\theta(m^{\frac{1}{d}})$ steps are needed for routing m packets, when m is not too small, with an additional log factor for CRCW. However, there is a difference for very sparse routing. When $m^{\frac{1}{d}}$ becomes smaller than $n^{\frac{1}{d+1}}$ the routing time cease to improve for CREW, while for CRCW there is no such break point. The $\Omega(m^{\frac{1}{d}})$ bound for both models is trivial as it follows from the isoperimetric inequality for the given geometry. As we shall see, an $\Omega(n^{\frac{1}{d+1}})$ bound for the CREW case is due to an inherent limitation in our ability to estimate threshold functions efficiently.

When buses and point-to-point communication are combined, a question that should be considered is whether none of these two media is redundant. In this respect, CRCW is quite different from CREW. For CREW, routing on meshes with buses can be done faster than routing without buses, and faster still using buses combined with point-to-point communication. In contrast, in the CRCW case, the addition of buses makes the point-to-point communication nearly redundant.

As indicated before, the main way to achieve faster results for CRCW is via Boolean threshold computations. We show how to compute the threshold-k Boolean function efficiently enough with CRCW buses.

We remark that for none of the proposed algorithms is the number of packets m, or even a bound on that number, assumed to be known in advance. Finally, we use the strongest assumption for lower bounds, namely, that information can be encoded and that transmission length may be arbitrary long. The algorithms themselves use no fancy encoding and packets are treated as atoms. The information being transfered is either packets or numbers in the range of the network size.

The rest of the paper is organized as follows: Section 2 contains basic definitions. In section 3 we present the key idea of the routing algorithm then we present the algorithms for CRCW and CREW for two dimensional mesh. In this section we also investigate the complexity of computing threshold functions (section 3.2) for the CRCW model. In section 4 we generalize the results for higher dimensional meshes. Section 5 presents the lower bounds for CREW.

2 Definitions and Notations

A bus in a network of processors, $G = (V, E)$, is a set $B \subseteq V$ that is thought of as a broadcast medium in which any single processor $v \in B$ can broadcast a message to which all other processors in B can listen. A network of buses is a graph $G = (V, E)$ with a set of buses $\mathcal{B} = \{B_1, B_2, ...\}$. We refer to the edges as point-to-point links. The networks that we deal with are synchronous. At each step every processor $u \in V$ may receive messages from its neighbors through its incident edges and transmissions through the buses B for which $u \in B$. Based on these data, processor u may send one message to each of its neighbors (on a point-to-point link) and a transmission on each of the buses it belongs to. An algorithm is a program resident in every processor $u \in V$. At every step it determines what messages and bus transmissions u should send. In the CREW case the algorithm should also guarantee that two or more processors should never broadcast on the same bus at the same step. In the CRCW model no such restriction is imposed and in the case of simultaneous attempts to broadcast on the same bus a special signal (noise) is heard by all who listen.

Definition 2.1 *A 1-1 routing problem for a network $G = (V, E)$ is to route m packets to their assigned destinations. It is assumed that each node $v \in V$ contains at most one packet and is the destination of at most one packet. The number of packets m, or any global knowledge on sources or destinations of the packets is not assumed to be known in advance.*

Let $G(n, d)$ denote the d-dimensional mesh with side length n and n^d points (processors). The network *as a mesh of buses* is equipped with dn^{d-1} 'buses' of size n each, one per each axis-parallel line. All the results we prove are for $d = O(1)$.

In the sequel we assume, for simplicity, that $n^{\frac{1}{d}}, n^{\frac{1}{d+1}}, m^{\frac{1}{d}}$ are integers (otherwise, m, n should be substituted with the smaller integers m', n', for which the above roots are integers. The order of magnitude is not changed and thus the results will hold).

3 Routing Algorithms

Our goal is to achieve routing time that is proportional to $O(m^{\frac{1}{d}})$ for m packets. The main idea of the algorithm is the following: Assume that there are at most $m^{\frac{1}{d}}$ packets on each bus along the first dimension (parallel to the first axis). In this case, the first coordinate of each packet can be corrected one packet after the other, using the buses along the first dimension. As this takes only one step for each packet, it can be done in $m^{\frac{1}{d}}$ steps. What remains are n separate routing problems on $(d-1)$-dimensional hyperplanes which may (in principle) be solved recursively. For this procedure to be efficient we need: (a) to rearrange the packets so that there are at most $m^{\frac{1}{d}}$ packets on each bus along the first dimension. (b) To control the number of packets on each $(d-1)$-dimensional hyperplane, so that the recursive routing on a lower dimensional planes can be done fast enough.

Let $G(n,d)$ be a d dimensional mesh. A line along the j-th dimension, $l_{\alpha,j}$ is a set of points whose projection along the j-th dimension is the single point α. A $(d-1)$-dimensional plane perpendicular to the j-th dimension, $H_{x,j}$, is the set of points whose j-th coordinate is x.

Definition 3.1 *Let $S \subseteq G(n,d)$ be a set of points in the d dimensional mesh, such that $|S| = m$. A $(d-1)$-dimensional plane $H_{x,i}$ is called* **"heavy"** *if it contains more than $m^{\frac{d-1}{d}}$ points of S, i.e. $|S \cap H_{x,i}| > m^{\frac{d-1}{d}}$. In that case the points of $S \cap H_{x,i}$ are said to be* **heavy** *for dimension i. If a plane (point) is not heavy, it is called* **light**.

A key lemma is the following,

Lemma 3.1 *Let $S \subseteq G(n,d)$ be an arbitrary set of points and let $H_{x,i}$ be any $(d-1)$-dimensional plane. Then $H_{x,i}$ contains at most $m^{\frac{d-1}{d}}$ points from S that are heavy for all dimensions.*

Proof: Let $|S| = m$. Then there are at most $m^{\frac{1}{d}}$ disjoint heavy planes perpendicular to each dimension. Otherwise the total number of points in S will exceed m. Fix a plane $H_{x,i}$, and let u be a point in $H_{x,i}$ that is heavy for all dimensions, possibly except i. Such a point $u \in H_{x,i}$ is determined by its intersection with $d-1$ heavy planes. As there are at most $m^{\frac{1}{d}}$ heavy planes in every dimension, then the total number of possible intersections is at most $m^{\frac{d-1}{d}}$. Hence, the number of such points on $H_{x,i}$ is at most $m^{\frac{d-1}{d}}$. ∎

The lemma suggests the following idea for a routing algorithm. We first present it in its simplest form for the two dimensional mesh, aiming for an $O(\sqrt{m})$ algorithm.

3.1 Routing in two dimensional meshes

Assume that we can bring all packets to the leftmost column with at most $\sqrt{m}$ packets at a site. Then the first coordinate of each packet can be corrected (bringing each packet to its correct column). Let S be the set of destinations. A column that is light with respect to S contains at most $\sqrt{m}$ packets. Thus packets on light columns can correct their second coordinate using their column bus in $\sqrt{m}$ steps. The remaining packets are those whose destinations are heavy with respect to columns. Assume that these packets can be brought to the bottom row. Then we repeat the process with rows and columns switching their roles. Lemma 3.1 guarantees that after the second trial all packets will reach their destinations, as no row will be heavy.

The outline above suggests the need of two types of operations: First we need to count the total number of packets, m, so to classify columns as light or heavy. In the case of CRCW, these operations will be reduced to the computation of Boolean threshold functions. In CREW these operations cannot be done as fast as for CRCW, and exact counting is done. The other operation is to bring the packets to the situation where the first coordinate can be corrected fast.

Along the sequel, we repeatedly move all packets that share a given bus through that bus. Scheduling the time in which each packet uses the bus will always be determined by indexing the packets according to a rank that will be computed beforehand by a threshold computation (CRCW) or by a counting operation (CREW).

3.2 Two dimensional CRCW routing

We begin by analyzing the complexity of computing the Boolean threshold function T_k^n.

Let each processor have an integer value x_i. The threshold-k function, T_k, is '1' if $\Sigma x_i \geq k$ and '0' otherwise.

Lemma 3.2 *[6] T_k can be computed in parallel for each line of $G(n,d)$ without using point-to-point communication in $O(k \cdot \log \frac{n}{k})$ steps.*

Proof: The original proof of [6] is for Boolean values. However, it works for integer values as well: processors on each line are split into $2k$ sets of equal size and in $2k$ steps the Boolean OR is computed on each of these sets (each processor transmits if its value is nonzero). If we get k positive answers we are done. Otherwise, we recursively continue with the processors in sets for which the answer was not '0' (at most half of the processors). This gives the following recursion $f(n) = 2k + f(\frac{n}{2}),\;\; f(2k) = 2k$ for which the solution is $f(n) = O(k \log \frac{n}{k})$.

It will be convenient to think of the integer value at a site as a Boolean value with multiplicity, since this will correspond to the existence of many packets. We note that if the answer to T_k is '0', then the algorithm also assigns each Boolean value in the input, a distinct rank $i < k$.

Theorem 3.1 *T_k can be computed on $G(n,2)$ in $O(\sqrt{k \log k} + \log n)$ steps without using point-to-point communication, and by $O(\min(\sqrt{k \log k} + \log n, n^{\frac{1}{3}}))$ steps using both buses and point-to-point communication.*

Proof: If $k^{\frac{1}{2}} \geq n^{\frac{1}{3}}$ then counting can be done in $O(n^{\frac{1}{3}})$ steps using point-to-point communication and buses, as in the CREW case (see section 3.4.1). Otherwise the algorithm is composed of the following phases:

1. All processors with non-zero value transmit on their row, i.e., an OR is computed on each row.

2. The number of rows containing a non zero value is computed and compared to k. This can be done in a straightforward manner in $O(\log n)$ steps. If this number is at least k the process is complete.

3. If the number of rows with a non zero value is at most $k-1$, we repeat the same operations on columns. Thus we reduce the $n \times n$ mesh to a $k \times k$ submesh (those rows and columns with a '1').

4. T_r is performed in each row simultaneously with $r = \sqrt{\frac{k}{\log k}}$. This can be done by Theorem 3.2 in $O(\sqrt{k \log k})$ steps.

5. The number of rows for which T_r is '1' is computed. If it is at least $\sqrt{k \log k}$ the process is completed. Otherwise, by repeating on columns (as we did before), the mesh is further reduced to a $\sqrt{k \log k} \times \sqrt{k \log k}$ submesh. In the latter case the task can be completed just by sequentially counting the '1' in each row and summing up.

The algorithm above takes $O(\sqrt{k \log k} + \log n)$ and uses only bus communication. We note that as before, in case that $\Sigma x_i < k$ the algorithm can also assign a distinct rank $i,\ i \leq k$ to each nonzero Boolean value.

We now turn back to the routing problem.

Theorem 3.2 *1-1 routing of m packets on CRCW $G(n,2)$ can be done in $O(m^{\frac{1}{2}} \log n)$ steps using only buses, and in $O(\min(m^{\frac{1}{2}} \log n, m^{\frac{1}{2}} + n^{\frac{1}{2}}))$ steps using point-to-point communication too.*

We note that m (or any bound on it) is not assumed to be known in advance.

Proof: Recall that our plan is first to project the packets to the plane and then to route them on light smaller dimensional planes. We first present the algorithm that uses only buses.

1. First m is estimated: starting from $m' = 1$, we compute $T_{m'}$ and keep doubling m' until the first time we get a '0' answer. Namely, we end up with an estimate m' for which $\frac{m'}{2} \leq m < m'$. By Theorem 3.1, this takes $O(\Sigma_{i=1}^{\log m}(2^{i/2}\sqrt{i} + \log n)) = O(\sqrt{m \log m} + \log m \log n) = O(\sqrt{m} \log n)$ steps.

2. Projection of the packets to the leftmost column is done in the following way: let m' be the estimate of the total number of packets and let $\ell = \sqrt{m'}$. T_ℓ is computed in parallel on each row in $O(\sqrt{m} \log n)$ steps. Packets in light rows (less then ℓ packets) are moved, one by one, to the left most column. This takes $\ell = O(\sqrt{m})$ steps. Then the process is repeated for columns. At the end of this phase Lemma 3.1 guarantees that each packet is either on the leftmost column or at the bottom row with $O(\sqrt{m})$ packets at a site.

3. Packets from the bottom row are brought to the leftmost column via the diagonal in $O(\sqrt{m})$ steps.

4. Now the packets are moved to their final destinations:

(a) The first coordinate is corrected using row buses in $O(\sqrt{m})$ steps (as there may be $O(\sqrt{m})$ packets at a site).

(b) T_ℓ is computed on each column to determine which columns are light (in $O(\sqrt{m}\log n)$ steps).

(c) Packets on light columns are brought to their final destinations along the columns in $O(\sqrt{m})$ steps.

(d) The remaining packets are brought back to the left most column and from there via the diagonal to the bottom row.

This phase is then repeated where rows and columns switch roles. Again, Lemma 3.1 guarantees that at the end of this phase each packet reaches its destination.

The whole algorithm takes $O(\sqrt{m}\log n)$ using only buses. The addition of point-to-point communication only makes threshold computation faster for $m \geq n^{\frac{1}{2}}$, as in this case counting on each line can be done in $O(n^{\frac{1}{2}})$ steps [1] (see also section 3.3). Thus we get $O(\min(\sqrt{m}\log n, m^{\frac{1}{2}} + n^{\frac{1}{2}}))$ steps using buses and point-to-point communication.

Remark: In fact the routing can be done in $m^{\frac{1}{2}} + n^{\frac{1}{3}}))$ steps by the same methods used for CREW (Theorem 3.3).

3.3 Two dimensional CREW routing

Routing in the CREW case is conceptually similar to the CRCW case. However, due to multiple broadcasts on the same bus, the CRCW threshold algorithm will not work. In fact threshold in CREW is as easy as counting the number of packets exactly. This in turn requires a different process to replace the projection (step 2) in case $m < n$ and gives a slower time bound for very sparse problems. On the other hand, our results in the CREW model are tight, in contrast to the CRCW case, where a gap of $\log n$ may exist between the upper and lower bounds for routing.

Theorem 3.3 *1-1 Routing of m packets in CREW $G(n,2)$ can be done in $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps.*

Proof: The algorithm has two main stages. First m is determined. If $m \geq n$ then exact counting on each row (or column) can be trivially done in $O(n^{\frac{1}{2}}) = O(m^{\frac{1}{2}})$ steps ([1]). Thus routing is the same as in CRCW where exact counting is done instead of threshold computations.

If $m < n$ then the packets are packed in a $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$ submesh, located at the corner of the 2-dimensional mesh. Then the packets are routed to their final destination as in the CRCW case.

We assume the ability to perform the following operations:

Counting- This operation determines the number of packets in $G(n,2)$. In addition, the counting also assigns each packet a distinct index in the range $1, ..., m$, where m is the total number of packets. Counting can be done in $O(n^{\frac{1}{3}})$ steps. The counting algorithm is described in Lemma 3.3.

projection Assume that m packets are placed in $G(n, 2)$, with at most one packet at each site. Then the packets can be projected to the bottom row, so that there will be at most $O(m^{\frac{1}{2}})$ packets at a site, in $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps. The projection algorithm is described in Lemma 3.4.

Packing Assume that $m < n$ packets are placed on the bottom row of $G(n, 2)$ with at most $O(m^{\frac{1}{2}})$ packets at a site. Then the packets can be moved to a $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$ sub-mesh so that there is at most one packet per site. The packing algorithm takes $O(m^{\frac{1}{2}})$ steps and is described in Lemma 3.5.

The routing algorithm is composed of the following phases:

Routing Algorithm - CREW

1. First m is determined by counting in $O(n^{\frac{1}{3}})$ steps. If $m \geq n$ then routing is done as in CRCW, with exact counting instead of threshold computations. Otherwise:
2. The projection operation is performed. This brings each packet to the bottom row with at most $O(m^{\frac{1}{2}})$ packets per site and takes $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps.
3. The packets are packed in a $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$ submesh located at the bottom left corner of the mesh. This is done in $O(m^{\frac{1}{2}})$ steps by the packing operation. Note that at the end of this operation, each site contains at most one packet. Consequently every column and every row are light.
4. Packets are moved to their final destination, exactly as in CRCW, except that counting on each row or column can be easily done as packets are located in a small segment. The process is as follows:
   (a) For each row of the packed submesh, in parallel, the first coordinate of each packet is corrected using a row bus. This takes $O(m^{\frac{1}{2}})$ steps as each row contains at most $O(m^{\frac{1}{2}})$ packets.
   (b) Counting is done on each column. As packets are in a segment of length $O(m^{\frac{1}{2}})$ on each column, this can be done trivially in $O(m^{\frac{1}{2}})$ steps (and in fact in $O(m^{\frac{1}{4}})$ steps [1]).
   (c) Packets on light columns are brought to their final destinations along the columns in $O(m^{\frac{1}{2}})$ steps.
   (d) The remaining packets are brought back to their position in the packed submesh.

   Then this process is repeated where rows and columns switch roles. Lemma 3.1 guarantees that at the end of this phase each packet reaches its destination.

The whole algorithm takes $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps.

Now we will describe in detail the counting and packing operations.

3.4 Counting, Projection and Packing

3.4.1 Counting

Lemma 3.3 *Assume that a set of packets is placed in $G(n,2)$ in an arbitrary way (possibly with more than one packet at some sites). Then, the number of packets can be computed in $O(n^{\frac{1}{3}})$ steps. As a side effect, each packet is also assigned a distinct index $i \leq m$, where m is the number of packets.*

Proof: The algorithm is a generalization of the counting algorithm for a line [1]. We divide $G(n,2)$ to $n^{\frac{4}{3}}$ sub-meshes of size $(n^{\frac{1}{3}} \times n^{\frac{1}{3}})$, and perform the following three phases:

1. The sum in each submesh is computed in $O(n^{\frac{1}{3}})$ (using only link connections) in a straightforward manner, by summing along each row and then summing partial sums. The sum of the (i,j) submesh is stored at the point $(i \mod n^{1/3}, 0)$ relative to its bottom left corner.
2. As each column contains $O(n^{\frac{1}{3}})$ sums, these sums can be transmitted to the bottom line, using the buses, in $O(n^{\frac{1}{3}})$ steps.
3. At this stage we are left with n partial sums on the bottom line. These values are summed in $O(\log n)$ steps in a straightforward manner. ∎

3.4.2 Projection

Lemma 3.4 *Assume that m packets are placed in $G(n,2)$, with at most one packet at each site. Then the packets can be projected to the bottom row, so that there will be at most $O(m^{\frac{1}{2}})$ packets at a site, in $O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps.*

Proof: The algorithm is composed of the following phases (see fig 1).

1. Let $r = \max(m^{\frac{1}{2}}, n^{\frac{1}{3}})$. $G(n,2)$ is divided into $(\frac{n}{r})^2$ sub-meshes of size $r \times r$. This creates a set of vertical bands in which projection is performed separately.
2. Sequential counting (using only link connections) is performed in every sub-mesh in $O(r)$ steps. The packets in each submesh are rearranged so as to form maximum number of full rows in each submesh. This can be done by any standard point-to-point routing algorithm for two dimensional mesh [14] in $O(r)$ steps.
3. At this stage every sub-mesh contains some *full rows* and at most one *incomplete row* with less than r packets (step 2 in figure 1). Full rows are counted and each full row is assigned a distinct ordinal number (this is done in a very similar way to the counting algorithm).

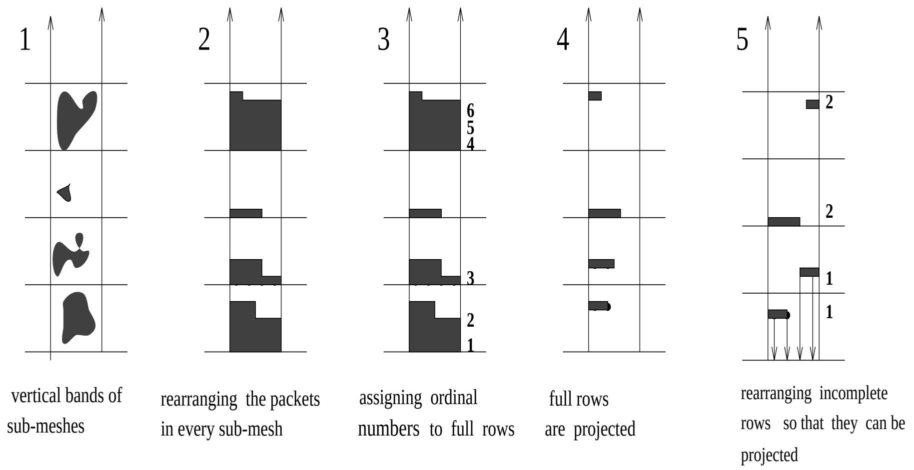

Figure 1: *Main stages of projection for the 2d mesh.*

Likewise, the incomplete rows are counted and assigned ordinal numbers too. The situation after this phase is depicted in step 3 of figure 1.

4. Full rows can be projected to the bottom row, one after the other, using their ordinal number. There are $\frac{m}{r} = O(m^{\frac{1}{2}})$ full rows; thus, this operation is completed in $O(m^{\frac{1}{2}})$ steps.

5. At this stage we are left with at most one incomplete row in each submesh. Since the number of packets in every incomplete row is known, the packets in the incomplete rows can be rearranged so that the incomplete rows of different submeshes form as many complete rows as possible (see step 5 of figure 1). At this stage, the situation is similar to that of the previous phase and thus it takes an additional $O(r)$ steps. Now there is at most one incomplete row in every vertical band and the process is terminated in one additional step.

Clearly the projection takes $O(r) = O(m^{\frac{1}{2}} + n^{\frac{1}{3}})$ steps. ∎

3.4.3 Packing

Lemma 3.5 *Assume that $m < n$ packets are placed in the bottom row of $G(n, 2)$ with at most $O(m^{\frac{1}{2}})$ packets at each site. Moreover, assume that m is known and that each packet is assigned a distinct index $i \leq m$. Then the packets can be packed in a $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$ submesh, with at most one packet per site, in $O(m^{\frac{1}{2}})$ steps.*

Proof: Each packet has an index $i \leq m$. We use this index to determine the final position that the packet should take in the $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$ submesh. Let $k = m^{\frac{1}{2}}$, expressing the index i in radix k defines two coordinates that then define the position of the packet in the submesh.

We describe the algorithm from the viewpoint of each individual packet. Assume a packet p is at $(x_1, 0)$ (on the bottom row), and is supposed to reach (a_1, a_2) (namely, its index $i = a_1 + a_2 k$).

- The packet moves to (x_1, i) where $i = a_1 + a_2 k$. Recall that initially there might be $O(m^{\frac{1}{2}})$ packets per site; thus $O(m^{\frac{1}{2}})$ steps are required. As no two packets share the same index i, there will be at most one packet at each row at the end of this phase.

- Each packet corrects its first coordinate. Namely, the packet in (x_1, i) moves to (a_1, i). As there is at most one packet on each row, this is done in one step.

- Each packet correct its last coordinate, using the last coordinate value for the schedule. Namely, the packet at (a_1, i) moves to (a_1, a_2) at step a_2. Note that there are no conflicts. Also, as $a_2 \leq m^{1/2}$ this is done in $O(m^{1/2})$ steps. ∎

4 Routing in higher dimensional meshes

The idea described at the beginning of section 3, in which the first coordinate is corrected and routing continue recursively on lower dimensional light planes, cannot be implemented as is, mainly because of accumulation of packets at the sites. For the two-dimensional case we had no problems; however, in three dimensions and up, we encounter a difficulty: Assume we start with m packets, at most one at a site in $G(n, 3)$. Once either projection or packing is done, the correction of the first coordinate may result in $\theta(m^{\frac{1}{3}})$ packets per site in every two dimensional plane. Thus we cannot directly apply the recursion, as our starting point of the algorithm is at most one packet per site.

Our aim is to reach a state in which after the first coordinate is corrected, we are left with at most one packet per site on each $(d-1)$-dimensional plane. As it turns out we can achieved this only if $m \leq n^{\frac{2d}{3}}$. For a larger m we divide the problem into $(m/n^{\frac{2d}{3}})$ disjoint problems which we solve one after the other.

We first describe the skeleton of the main part of the routing algorithm. It will be the same for both CRCW and CREW. We assume that a bound on the number of packets m, is known, and that each packet has a distinct rank in the range $1, .., m$.

Along the sequel, we use the $O(.)$ notation in several places, indicating that a leading factor that is independent of m and n but is dependent on d. This factor can be easily determined; however, the description is simplified considerably with this notation.

Algorithm Routing(m,d) The algorithm starts with at most $m \leq n^{\frac{2d}{3}}$ packets in $G(n, d)$, with at most one packet per site. Furthermore, we assume that *m is known and each packet has a distinct rank in the range* $1, ..., m$. The algorithm is recursive, where in the base case of $d = 2$ we apply the algorithms of section 3.1. The proposed algorithm, as depicted in figure 2, is composed of the following phases:

1. **Packing** The packets are packed in a submesh $G(m^{\frac{1}{d}}, d)$, located at the leftmost corner of $G(n, d)$, with at most one packet per site. This procedure is in its self recursive. It takes $O(m^{\frac{1}{d}} \log n)$ steps for CRCW and $O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$ steps for CREW. The packing operation is described in Lemma 4.3 (CRCW) and Lemma 4.5 (CREW), see also and figure 3.

2. **Routing from a packed submesh** For each dimension $r = 1, ..., d$, we now attempt to correct the r-th coordinate and proceed recursively on light planes. By Lemma 3.1 this will do. More formally, routing from a packed submesh uses the following steps:
   Repeat for each of the dimensions $i = 1, ..., d$:

   (a) Packets correct their i-th coordinate using the buses. As there are only $O(m^{\frac{1}{d}})$ packets on each bus (with a natural ordering on packets), this takes $O(m^{\frac{1}{d}})$ steps.

   (b) Counting is performed on every $(d-1)$-dimensional plane that is perpendicular to the i-th dimension. Each such plane is a $(d-1)$-dimensional mesh of side length n; however, note that the packets in each such plane are in fact in a $(d-1)$-dimensional submesh of side length $m^{\frac{1}{d}}$. Thus this counting is done in $O(m^{\frac{1}{d} \cdot \frac{1}{d}})$ steps even for CREW. Every plane is marked as light or heavy according to the number of packets destined to it. There might be, however, up to $O(m^{\frac{1}{d}})$ packets per site, making it impossible to proceed directly by induction. The only purpose of the last two phases is to classify planes as light and heavy.

   Our purpose now, in the following two steps, is to bring all packets to the bottom plane with at most one packet per site.

   (c) All packets are brought back to their original locations in the packed submesh.

   (d) As $m < n^{\frac{2d}{3}} < n^{d-1}$, each packet has a distinct rank in the range $1, .., n^{d-1}$. We use this rank as an intermediate destination, r, on the bottom plane. For each $(d-1)$-dimensional plane $H_{x,i}$, we route each packet to its intermediate destination in $H_{x,i}$. Namely, a packet in $H_{x,i}$ that has intermediate destination $r = (a_1, .., 0, a_{i+1}, ..., a_{d-1})$ is routed in $H_{x,i}$ to $(a_1, .., i, a_{i+1}, ..., a_{d-1})$. This routing can be done directly, but in particular it can be done recursively inside each $(d-1)$-dimensional plane $H_{x,i}$ by calling the "routing from packed submesh". Note that initially each site contains at most one packet (in the packed submesh). Also note that there is an a-priori bound, m', on the number of packets in each such plane, namely $m' \leq m^{\frac{d-1}{d}}$ and since $m \leq n^{\frac{2d}{3}}$ we have $m' \leq n^{\frac{2d}{3}\frac{d-1}{d}} \leq n^{\frac{2(d-1)}{3}}$. Thus, the condition for calling routing on the $(d-1)$-dimensional planes with the bound m' is met.

   At the end of this phase there is at most one packet on each line $l_{\alpha,i}$.

   (e) Packets correct their i-th coordinate. This is done using the buses in one step as there is at most one packet on each bus along the i-th dimension.

   As in the situation after step 2b, packets now are at the right plane in dimension i;

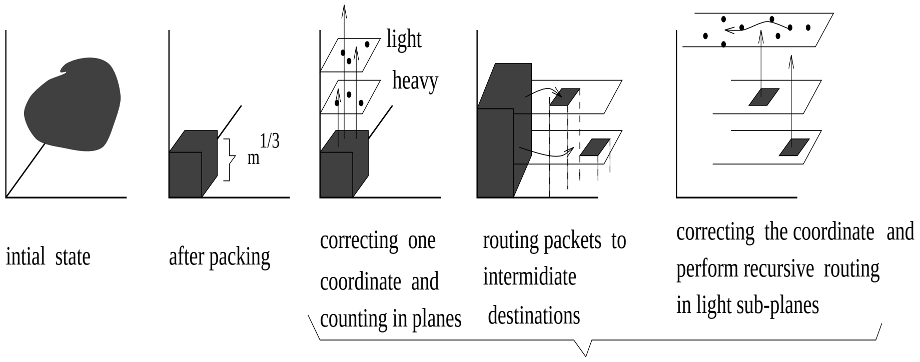

Figure 2: *Main stages of the routing algorithm for the 3d mesh.*

however, now there is at most one packet at a site. Moreover, the number of packets at each $(d-1)$-dimensional plane is known (along with their rank) as a result of phase 2b.

(f) At this point routing on the light $(d-1)$-dimensional planes perpendicular to the i-th dimension is done recursively.

Theorem 4.1 *Assume that at most $m \le n^{\frac{2d}{3}}$ packets are in $G(n,d)$ with at most one packet per site. Moreover, assume that m is known and every packet has a distinct rank in the range $1, ..., m$. Assume also that packing can be done in time as stated in phase 1 of the algorithm above. Then the algorithm is correct and ends in $O(m^{\frac{1}{d}} \log n)$ for CRCW and $O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$ for CREW.*

Proof: Lemma 3.1 guarantees that if for each dimension i, the i-th coordinate is corrected and then packets are routed in light planes perpendicular to the i-th dimension, then the entire routing is completed. The goal of the first packing phase is to enable us to count quickly and to reach the state in which after correcting a coordinate, there is at most one packet per site, which in turn is a precondition for calling the routing recursively on a $(d-1)$-dimensional plane. Assuming this is done correctly, phase 2d rearranges the packets so that there is only one packet on each bus along the i-th dimension. Then in phase 2e, packets correct their i-th dimension. Since after phase 2d, there is at most one packet on each bus along the i-th dimension, then after correction of the i-th coordinate there will be at most one packet per site. Phase 2f is then the recursive call on light planes. As noted before, there is at most one packet per site, and there is an a-priori bound $m' \le m^{\frac{d-1}{d}}$ on the number of packets in light planes. This implies $m' \le n^{\frac{2(d-1)}{3}}$, as required by the recursive call.

The complexity for CRCW is: $time(packing) + d \cdot (O(m^{\frac{1}{d}}) + 2 \cdot time(Routing(m^{\frac{d-1}{d}}, d-1)))$. Using induction (on d) and substituting we get: $time(Routing(m,d)) \le O(m^{\frac{1}{d}} \log n)$.

For CREW, let $f(m,d)$ denote the time of routing from a packed submesh, then: $f(m,d) = d(O(m^{\frac{1}{d}}) + 2f(m^{\frac{d-1}{d}}, d-1))$ solving using $f(m,2) = O(m^{\frac{1}{2}})$ we get $f(m,d) = O(m^{\frac{1}{d}})$. Thus

$routing(m,d)$ takes $O(m^{\frac{1}{d}}) + time(packing) = O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$. ∎

We note that for CRCW, for large m we can use counting as in CREW which brings the term $m^{\frac{1}{d}} + n^{\frac{1}{d+1}}$ as an upper bound too.

It remains to be shown how to estimate a bound on m and to rank the packets. This is done by first performing counting (CREW) or threshold computation using the doubling method (CRCW). Thus we get:

Theorem 4.2 *Assuming the ability to do packing in time as stated in phase 1 of the algorithm Routing(d,m) then 1-1 routing of at most m packets, can be done in $O(\lceil (m/n^{\frac{2d}{3}}) \rceil \min(m^{\frac{1}{d}} \log n, m^{\frac{1}{d}} + n^{\frac{1}{d}}))$ steps for CRCW and in $O(\lceil (m/n^{\frac{2d}{3}}) \rceil (m^{\frac{1}{d}} + n^{\frac{1}{d+1}}))$ steps for CREW.*

Proof: First m is estimated assigning ranks to every packet. This is done in $O(\min(\ (m \log m)^{\frac{1}{d}}\ ,\ n^{\frac{1}{d+1}}))$ for CRCW by Lemma 4.2 below, and in $O(n^{\frac{1}{d+1}})$ steps for CREW (Theorem 4.4).

If $m \geq n^{\frac{2d}{3}}$, then the rank is used to divide the packets into $(m/n^{\frac{2d}{3}})$ sets with at most $n^{\frac{2d}{3}}$ packets in each. Then $Routing(d,m)$ is called on each set. ∎

4.0.4 Counting and Packing for CRCW

Lemma 4.1 *T_k can be computed on $G(n,d)$ in $O(\min(\ (k \log k)^{\frac{1}{d}}, n^{\frac{1}{d+1}}))$ steps.*

Proof: The algorithm is a straightforward generalization of the algorithm in Theorem 3.1. The minimum with $n^{\frac{1}{d+1}}$ derives from the fact that exact counting can be done in that time even for CREW. ∎

Lemma 4.2 *An estimation m' of the actual number of packets m, can be computed in $O(\min(\ (m \log m)^{\frac{1}{d}}\ ,\ n^{\frac{1}{d+1}}))$ steps, such that $\frac{m}{2} \leq m' \leq m$. Moreover, each packet is assigned a distinct rank in the range $1, ..., m'$.*

Proof: The algorithm is a straightforward generalization of the algorithm for the two-dimensional mesh that is described in the first phase of the algorithm in Theorem 3.2.

Lemma 4.3 *Assume that m is a known upper bound on the number of packets, and that each packet packet has a distinct rank in the range $1, ..., m$, then packing in the CRCW mesh can be done in $O(m^{\frac{1}{d}} \log n)$ steps.*

Proof:

The algorithm is recursive. If $d = 1$ then the rank is used to send the packets to the packed subinterval.

For $d \geq 2$ the algorithm is composed of the following phases:

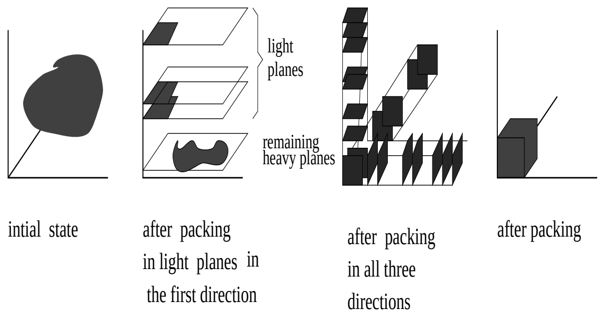

Figure 3: *Main stages of packing for the 3d mesh.*

1. Let $\ell = m^{\frac{d-1}{d}}$. For $i = 1, ..., d$ the following is repeated:
   - T_ℓ is computed in every $(d-1)$-dimensional plane that is perpendicular to the i-th dimension. This can be done in $O(\min((\ell \log \ell)^{\frac{1}{d-1}}, n^{\frac{1}{d}})) = O(\min((m \log m)^{\frac{1}{d}}, n^{\frac{1}{d}}))$ steps.
   - Packets in light planes (that are not already packed) are packed recursively by calling Packing on each $(d-1)$ light plane. At the end of this phase, every packet has at least $d-1$ coordinates in the right range $(0, ..., m^{\frac{1}{d}} - 1)$. This is because Lemma 3.1 asserts that as the above is repeated for each dimension, then there are no heavy packets for the last dimension. One iteration of this phase takes $time(pack(d-1, m^{\frac{d-1}{d}}))$. The situation at this point is depicted in figure 3 (third step).

2. At this point the packets can be partitioned into at most d sets, $S_i, \quad i = 1, .., d$, where all packets in S_i have all but the i-th coordinate in the right range. We now pack each S_i to its own submesh; these submeshes can then be trivially embedded in a d times larger submesh, in the lower leftmost corner. We show how to rearrange S_1 into a proper submesh and describe it for $d = 2$ (The generalization for higher dimensions is straightforward).

   The sites in S_1 are in the submesh defined by $\{1, ..., m^{\frac{1}{2}}\} \times \{1, ..., n\}$. We divide this "band" into $\frac{n}{\sqrt{m}}$ submeshes each of size $m^{\frac{1}{2}} \times m^{\frac{1}{2}}$. Now we proceed exactly as from phase 2 in the projection for two dimensions, in Lemma 3.4. The only difference is that full rows are projected one on top of the other rather than to the bottom row. This phase takes $O(m^{\frac{1}{2}})$ steps for $d = 2$ and $O(m^{\frac{1}{d}})$ for the d dimensional case.

The whole algorithm takes: $time(pack(m, d)) = d \cdot (time(T_\ell) + time(pack(m^{\frac{d-1}{d}}, d-1))) + O(m^{\frac{1}{d}}) = O((m \log m)^{\frac{1}{d}})$. ∎

4.0.5 Counting, Packing for CREW

Lemma 4.4 *Assume that a set of packets is placed in $G(n, d)$ in an arbitrary way (possibly with more than one packet at some sites). Then, the number of packets can be computed in $O(n^{\frac{1}{d+1}})$*

steps. As a side effect, each packet is also assigned a distinct index $i \leq m$, where m is the number of packets.

Proof: The algorithm is a straightforward generalization of the algorithm given in the proof of Theorem 3.3.

Lemma 4.5 *Assume that m is a known upper bound on the number of packets, and that each packet packet has a distinct rank in the range $1, ..., m$, then packing in the CREW mesh can be done in $O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$ steps.*

Proof: If $m \geq n$, then we can directly count in $(d-1)$-dimensional planes in $O(n^{\frac{1}{d}}) = O(m^{\frac{1}{d}})$. Thus the algorithm is exactly as for CRCW.

If $m < n$ we first project to the bottom plane and then we pack. Both these operations are straightforward generalizations of the corresponding projection (Lemma 3.4) and packing (Lemma 3.5) for $d = 2$. These projection and packing take $O(m^{\frac{1}{d}} + n^{\frac{1}{d+1}})$ and $O(m^{\frac{1}{d}})$ respectively. ∎

5 Lower bounds for CREW

As in the CRCW case, a trivial lower bound of $\Omega(m^{\frac{1}{d}})$ for routing of m packets can be easily obtained by packing the packets in a sub-mesh with destinations outside this sub-mesh. This lower bound holds for algorithms that use only buses, only point-to-point links or both. Furthermore, this lower bound holds even if global information is known in advance and off-line computation is allowed. However, this gives nothing for $m = O(1)$. We prove here a tight lower bound in terms of the mesh size n rather than m.

Here we present two tight lower bounds, one for the case when only buses are used, and the other for the case when both buses and point-to-point connections are used. Both lower bounds are for the strongest possible paradigm of communication, where there are no restrictions on the length or type of the information that is being transferred. In particular, the algorithm may concatenate or encode information in an arbitrary way.

Let us consider first the case where point-to-point communication is not used.

Theorem 5.1 *1-1 routing in a d-dimensional mesh of CREW buses requires at least $\frac{n}{d} - 1$ steps when only bus connections are used.*

For simplicity we describe the lower bound for the two dimensional case.

Proof: We prove that even if it is known in advance that there are only 2 packets, then at least $\frac{n}{2} - 1$ steps are required. The following adversary argument is used:

Let A be an algorithm that routes 2 packets in t steps. Consider the first step of A, as no two processors may broadcast on the same bus, an adversary will not assign packets to the processors that do broadcast. Since for each of the $2n$ buses, only one processor may broadcast, then the adversary loses at most $2n$ processors as candidates for originating packets. Intuitively the adversary will try to carry on with this policy as long as possible. Thus, as long as there is a non empty set of processors that are candidates for originating packets, the algorithm can not yet terminate. There is, however, a slight flaw in this argument. In fact, two processors (or more) may, in general, want to broadcast on the same bus at a certain step: it may be the case that several processors are assigned to broadcast on a given bus, at a given step, under the condition that their packets have a predetermined destination d. In this case, the CREW paradigm would not be violated as this would never actually happen for a $1-1$ routing problem. To overcome this difficulty, the adversary will refrain from assigning certain destinations as targets to the (future) assigned packets; that is, she will delete from the list of possible target destinations those destinations that occur in a situation as that described above.

Formally, we call a broadcast at step t a **"positive broadcast"** if it was sent by either a processor which is an origin of a packet, or a processor that received a positive broadcast in a previous step. The adversary will try to keep the condition that no processor transmits a positive broadcast in the first t steps. Let $V^t \subset V(G)$ be the set of processors that are still candidates to be origins of packets after step t. Let $D^t \subseteq V(G)$ denote the set of all sites that are candidates for being destinations for packets after step t. Namely, V^t, D^t are sets for which any 1-1 mapping of two processors from V^t to D^t is consistent with the first t steps of the algorithm. We will prove by induction on t that as long as $t < \frac{n}{2} - 1$ then $|V^{t-1}| \geq n^2 - 2nt$ and $|D^{t-1}| \geq n^2 - 2nt$, and no positive broadcast has been made. Note that these conditions hold for $t = 0$ (just before the first step) with $D^0 = V^0 = V(G)$.

Assume now that D^{t-1}, V^{t-1} are defined and that the above conditions hold. Let us focus on the t-th step. Let $X \subseteq V^{t-1}$ be the set of all processors that may transmit a positive broadcast on any of the consistent inputs (any $1-1$ mapping of two processors from V^{t-1} to D^{t-1}) on a certain given bus. We claim that either $|X| = 1$ or that there is a predetermined destination $d \in D^{t-1}$, so that every processor in X broadcasts only if its packet has destination d. Otherwise we may assume that p_1, p_2 are two processors that are scheduled to transmit on the bus if they have packets with destinations $d_1 \neq d_2$ respectively. As $\{p_1, p_2\} \subseteq V^{t-1}$ and $\{d_1, d_2\} \subseteq D^{t-1}$, and we have assumed that there were no positive broadcasts so far, then the assignment of a packet to p_i with destination d_i, $i = 1, 2$ is consistent with the transmissions so far. However, in the t step, both p_1 and p_2 attempt to broadcast, violating the CREW paradigm.

Thus, for each of the $2n$ buses, there may be at most one predetermined destination so that several processors may attempt to broadcast if they possess packets with that destination. Let E be the set of these 'bad' destinations. The adversary deletes E from D^{t-1}, i.e., $D^t = D^{t-1} \setminus E$. Consider now all processors that might broadcast on packets that have destinations in D^t. According to the above claim there is at most one such processor per bus. Let P be the set of all these processors; the adversary deletes P from the set of candidates V^{t-1}, that is, $V^t = V^{t-1} \setminus P$. As $|E| \leq 2n$ and $|P| \leq 2n$; it follows that $|V^t| \geq |V^{t-1}| - 2n \geq n^2 - 2nt$. For D^t the result is the same. Moreover, no positive broadcast has been made in the first t steps. This implies that for any $D' \subseteq D^t$, $I' \subseteq I^t$ with $|D'| = |I'| = 2$, any $1-1$ mapping of destinations to processors $\varphi : D' \longrightarrow I'$ is consistent with the transmissions made in the first t steps.

We conclude that the adversary can carry on with this policy for any $t < \frac{n}{2} - 1$ steps, such that not a single positive broadcast has been made during these steps. In particular the processors holding the packets to be routed will not make a single transmission. ∎

We note that if the number of packets, m, is known in advance then a similar argument implies a lower bound of $\frac{n}{2} - \frac{m}{2n}$ (for $m \geq 2$).

Next we prove a lower bound for 1-1 routing on a mesh of buses when both point-to-point and bus communication are used.

Theorem 5.2 *1-1 routing in a d-dimensional mesh of buses requires at least $\Omega(n^{\frac{1}{d}})$ steps, when both point-to-point and buses are used.*

Proof: Again, for simplicity, we present the proof for $d = 2$. We use here essentially the same method as in the proof of Theorem 5.1, however, we have to take into account the possible effect of the point-to-point communication. Intuitively, this is done by fixing a distance of at least $n^{\frac{1}{3}}$ between processors that are given packets by the adversary. Thus "positive" information cannot "escape" a circle of radius $n^{\frac{1}{3}}$ without using buses.

Formally, we partition $G(n, 2)$ into $n^{\frac{4}{3}}$ sub-meshes of size $n^{\frac{1}{3}} \times n^{\frac{1}{3}}$ each. The adversary will assign packets only to processors at the centers of sub-meshes.

We define a broadcast or a point-to-point message at step t to be **"positive"** if it was sent by either a processor from which a packet originates or a processor that received a positive broadcast/message in a previous step. We say that a processor is positive if it received or transmitted a positive message or broadcast. For each positive processor p we can trace back a center of a sub-mesh that is the **source** of the sequence of positive messages and broadcasts that made p positive. We call this center "a source" of p (if there is more than one such center then we designate an arbitrary one as the source).

The adversary's goal is to prevent positive broadcasts for as long as possible. She will do so by never assigning packets to processors in those sub-meshes in which there are processors attempting to transmit a positive broadcast on a bus. There is of course no way to avoid positive messages on point-to-point links. Thus, the 'real' information will be spread only by using point-to-point communication which limits the distance that information can reach. For convenience the destinations will also be on centers of sub-meshes.

Let V^t be the set of sub-meshes whose centers are still candidates to be origins of packets after step t. Let D^t be the set of sub-meshes whose centers are still candidates to be the destinations of packets after step t. The adversary will try to hold to the condition that no processor in any sub-mesh $m \in V^t$ will transmit a positive broadcast in the first t steps. Moreover, for any set of 2 sub-meshes $I \subseteq V^t$, and any set of two destinations $D \subseteq D^t$, any $1-1$ assignment of the destinations D to the processors I (i.e centers to centers) will be consistent with the first t steps of the algorithm.

We will assume by induction, that D^{t-1}, V^{t-1} are defined, that the above conditions hold, and that $|V^{t-1}| \geq n^{\frac{4}{3}} - 2nt$ and $|D^{t-1}| \geq n^{\frac{4}{3}} - 2nt$. Clearly, this holds for D^0 and V^0 that are the set of all $n^{\frac{4}{3}}$ sub-meshes.

Fix a bus B. Let X be the set of all processors that may transmit a positive broadcast on B, at step t, on any of the consistent inputs so far. As long as $t < n^{\frac{1}{3}}$, and since there were no positive broadcasts so far, each of the processors in X has the center of its sub-mesh as its only possible source of a sequence of positive messages. As in the proof of Theorem 5.1 we claim that either all processors in X are in the same submesh (namely they correspond to one packet) or there is a predetermined destination $d \in D^{t-1}$ (which is the center of a sub-mesh $m \in D^{t-1}$), so that every processor in X broadcasts only if its source has a packet with destination d. The adversary removes all "bad" destinations D' (at most $2n$) from D^{t-1}, that is $D^t = D^{t-1} - D'$.

Consider now all processors that might broadcast with inputs consistent with D^t. There is at most one such processor per bus. Let P be the set of all sub-meshes that contain these processors. The adversary deletes P from the set of candidates, $V^t = V^{t-1} \setminus P$, eliminating complete sub-meshes in order to prevent broadcasts from processors that could have received a packet through a sequence of point-to-point communications in the first $t-1 \leq n^{\frac{1}{3}}$ steps. As $|P| \leq 2n$ it follows that $|V^t| \geq |V^{t-1}| - 2n \geq n^{\frac{4}{3}} - 2nt$ and similarly for D^t. Moreover, no positive broadcast has been made in the first t steps. This also implies that for any $D' \subseteq D^t$, $I' \subseteq I^t$ with $|D'| = |I'| = 2$ any $1-1$ mapping of destinations to processors $\varphi : D' \longrightarrow I'$ is consistent with the transmissions made in the first t steps. Thus, as long as $t < n^{\frac{1}{3}}$, this process can be carried on while there are no positive broadcasts. This implies that the routing cannot be completed and thus the routing time must be $\Omega(n^{\frac{1}{3}})$. ∎

6 Conclusions

In this paper, we have demonstrated that the combination of buses and point to point communication improves the performance of routing problems on the mesh topology. Both CREW and CRCW models were analyzed. We showed that in both cases essentially optimal algorithms can be constructed, so that the 'spareness' of the problem is fully exploited in order to accelerate the routing time.

The difference between the two models in respect to routing lies in the ability to compute thresholds. The improved ability in the CRCW case makes it possible to accelerate the routing time in accordance to the decrease in the number of packets even for very low load. In the CREW model this ability to make use of the 'sparseness' of the problem is limited by the ability to count. We proved that in fact the complexity of counting is a lower bound on the routing. On the other hand, our lower bound on routing implies that our upper bounds for counting are tight too.

Another issue concerning the practical application is the buffer size. In the algorithms that we have presented the buffers are quite large ($\theta(m^{1/d})$). A closer look reveals that the maximum buffer size can be decreased to a constant depending only on d for $m \leq n^{\frac{2d}{3}}$. In essence, the reason for this is that we reduce 1-1 routing in dimension d to 1-1 routing in dimension $d-1$. Along the reduction, packets accumulated at sites only for counting, but this could be done without actually moving the packets at all. Instead, token that represent the packets can be counted.

Open problems:

- The hidden factor in our algorithms show a dependence on d of the sort $d!$. We suspect that this dependence can be made to be polynomial in d. The question of the dependence on d is left open; In particular, the lower bound question is quite interesting.
- The algorithms in the CRCW case are optimal up to $\log n$ factor. Can this be improved ?
- The question of the exact complexity of computing threshold-k in the CRCW model, as well as other families of Boolean functions, is interesting in its own right and has not been fully determined.

References

[1] Y. Afek, G. Landau, B. Schieber, and M. Yung. The power of multimedia: Combining point-to-point and multiaccess network. *Information and Computation*, 84(1), Jan 1990.

[2] A. Aggarwal. Optimal bounds for finding maximum on array of processors with k global buses. *IEEE Trans. on Comp.*, 35:62–64, Jan 1986.

[3] F. Meyer auf der Heide and H. Thien Pham. On the performance of networks with multiple buses. In *STACS92: Springer-Verlag*, pages 97–108, 1992.

[4] Andrew G. Bale, John Litt, and Cliff J. Pavelin. The AMT DAP 500 system. In T. J. Fountain and M. J. Shute, editors, *Multiprocessor Computer Architectures*, pages 155–184. North-Holland, 1990.

[5] A. Bar-Noy and D. Peleg. Square meshes are not always optimal. In *28th Annual ACM Symp. on Parallel Algorithms and Architectures*, pages 138–147, 1989.

[6] Y. Ben-Asher and I. Newman. Decision trees with boolean threshold queries. *Journal of Computer and System Sciences*, 51(3):495–502, 1995.

[7] S. H. Bokhari. Finding maximum on an array processor with a global bus. *IEEE Trans. on Comp.*, 33:1984, Feb 1984.

[8] Y-C. Chen, W-T. Chen, , and G-H. Chen. Efficient median finding and its application to two-variable linear programming on mesh connected computers with multiple broadcasting. *Journal of Parallel and Distributed Computing*, 15:79–84, 1992.

[9] Charles M. Fiduccia. Bused hypercubes and other pin-optimal networks. *IEEE Trans. Parallel & Distributed Syst.*, 3(1):14–24, Jan 1992.

[10] Ariel J. Frank, Larry D. Wittie, and Arthur J. Bernstein. Multicast communication on network computers. *IEEE Software*, 2(3):49–61, May 1985.

[11] Jean-Luc Gaudiot, Rex W. Vedder, George K. Tucker, Dennis F inn, and Michael L. Campbell. A distributed VLSI architecture for efficient signal and data proce ssing. *IEEE Trans. Comput.*, C-34(12):1072–1087, Dec 1985.

[12] Wolfgang K. Giloi. SUPRENUM — a MIMD/SIMD supercomputer for numerical applications. In G. Paul and G. S. Almasi, editors, *Parallel Systems and Computation*, pages 165–178. North-Holland, 1988.

[13] K. Hwang, M. Dubois, D. K. Panda, S. Rao, S. Shang, A. Uresin, W. Mao, H. Nair, M. Lytwyn, F. Hsieh, J. Liu, S. Mehrotra, and C. M. Cheng. OMP: A RISC-based multiprocessor using orthogonal-access memories and multiple spanning buses. In *Intl. Conf. Supercomputing*, pages 7–22, Jun 1990.

[14] F.T. Leighton. *Introduction to parallel algorithms and architectures.* Morgan Kaufmann publishers, 1991.

[15] J. Y-T. Leung and S. M. Shende. Packet routing on square meshes with row and column buses. In *IEEE Symposium on Parallel and Distributed Processing, Dallas, Texas*, pages 834–837, 1991.

[16] V.K. Prasanna-Kumar and C.S. Raghavendra. Array processor with multiple broadcasting. *Journal of Parallel and Distributed Computing*, 4:173–190, 1987.

[17] S. Rajasekaran. Mesh connected computers with fixed and reconfigurable buses: packet routing, sorting and selection. In *proc. 1st European Symp. on Algorithms*, September 1993.

[18] J. F. Sibeyn, M. Kaufmann, and R. Raman. Randomized routing on a mesh connected array with buses. In *1st Annual European Symposium on Algorithms, Springer-Verlag LNCS series v.726, pp. 333-344*, 1993.

[19] Q. F. Stout. Mesh-connected computers with broadcasting. *IEEE Transactions on Computers*, 32:826–830, 1983.

[20] Q. F. Stout. Meshes with multiple buses. In *IEEE Symposium on Foundations of Computer Science*, number 27, pages 264–273, 1986.

[21] T. Suel. Routing and sorting on meshes with row and column buses. In *8th International Parallel Processing Symposium*, pages 411–417, 1994.

[22] Shreekant Thakkar et al. New directions in scalable shared-memory multiprocessor architectures. *Computer*, 23(6):71–83, Jun 1990.