2次元メッシュバス上での高速な確率ラウティングアルゴリズム

田嶋 聡* 岩間 一雄** 宮野 英次*

* 九州大学大学院システム情報科学研究科 ** 京都大学工学部

メッシュバス計算機 (MBUS) とは，$n \times n$ のプロセッサ間を n 本の行バスと n 本の列バスで結合した並列計算機モデルである．本稿では確率ラウティングアルゴリズムを提案し，このアルゴリズムは高い確率で $1.46875n + o(n)$ ステップ時間のラウティングを行なうことを示す．決定性ラウティングの場合には，$1.5n$ 時間のラウティングアルゴリズムが知られているが，本結果は，乱数を用いることで $1.5n$ 時間より早くラウティングを行なうことが可能であることを示している．

Efficient Randomized Rouitng Algorithm on the Two-Dimentional Mesh of Buses

Satoshi Tajima*, Kazuo Iwama** and Eiji Miyano*

*Department of Computer Science and Communication Engineering Kyushu University

**Department of Information Science Kyoto University

The mesh of buses (MBUSs) is a parallel computation model which consists of $n \times n$ processors, n row buses and n column buses but no local connections between neighboring processors. In this paper, we present a randomized permutation routing algorithm that runs in $1.46875n+o(n)$ steps on MBUSs with high probability. Our result is interestng since the $1.5n$ upper bound shown in [Iwama et. al. J of Algorithms, 1996] seems to be the best possible for deterministic routing.

1 Introduction

The two dimensional mesh is widely considered to be a promising parallel architecture in its scalability. In this architecture, processors are naturally placed at intersections of horizontal and vertical grids, while there can be two different types of communication links: The first type is shown in Figure 1(a). Each processor is connected to its four neighbors and such a system is called a *mesh-connected computer* (an *MC* for short). Figure 1(b) shows the second type: Each processor is connected to a couple of (row and column) buses. The system is then called a *mesh of buses* (an *MBUS* for short).

Routing is a basic form of communication among the processors: The input is given by n^2 packets that are initially held by the $n \times n$ processors, one for each. Routing requires that all n^2 such packets be moved to their final destinations. In the case of MCs, a $2n-2$ lower bound easily comes from the physical distance between farthest two processors without any condition. Also, a $2n-2$ upper bound can be achieved by using the most rigid, dimension-order path strategy [6]. On the other hand, in the case of MBUSs, the dimension-order path strategy does not work so optimally since the strategy must take $2n$ steps, and the $\lceil 1.5n \rceil$ upper bound seems to be the best possible [3]. That is a tight bound for permutation routing if we impose so-called "strongly oblivious" condition [4], but only a $(1-\varepsilon)n$ lower bound without any condition is known [3].

The main purpose of this paper is to decreases the $\lceil 1.5n \rceil$ upper bound by allowing randomization. We prove that any permutation can be routed over the MBUSs in $1.46875n + o(n)$ steps with high probability. Our randomized algorithm consists of the following two stages. In the first stage, each packet is moved vertically to its correct position with respect to row with probability $\frac{1}{2}$, or horizontally to its correct position with respect to column also with probability $\frac{1}{2}$. Namely, one can expect that, in total, $\frac{n^2}{2}$ packets are moved to their correct row positions and the other $\frac{n^2}{2}$ to their correct column positions. The basic strategy on how each packet chooses the direction, row or column, is to make use of the collision of two or more packets on a bus: Consider some four processors on a single row, say, P_1, P_2, P_3 and P_4. (1) P_1 and P_2 try to write their initial packets with probability $\frac{1}{2}$ on the row bus. (2-1) If only one of the two processors moves the packet, then P_3 moves its packet. (2-2) If both try to write their packets, i.e., there is a collision, then again P_1 moves its packet and then P_4 moves the packet. (2-3) If neither P_1 nor P_2 tries to write the packets, then first P_2 and then P_4 move their packets. Thus, exactly two of the four processors can succeed in moving their initial packets. The remaining two processors will move their packets using the column bus later. In the second stage, every packet moves to its final position also by using randomization, but the idea is much more tricky. See Section 3 for more details.

One can naturally think of the model which is equipped with both buses and local connections (*MCs with buses*). The known lower and upper bounds for this model are $0.691n$ [2] and $(1+\varepsilon)n + o(n)$ [5], respectively. Recall that the $(2n-2)$-lower bound for point-to-point communication models comes from the diameter; even if randomization is allowed, this lower bound does not change. However, the situation is completely different for bus communication; it should be noted that the lower bound for MBUSs essentially comes from the fact that many packets gather at some single bus. If we allow randomization, then one can expect that the addition of buses becomes especially beneficial since randomization may remove such high packet-congestion. An interesting question is whether or not we are able to decrease the upper bound for MCs with buses by using the similar randomized approach as above.

2 Model and Result

An MBUS consists of n^2 processors, $P_{i,j}$, $1 \leq i,j \leq n$, and n row and n column buses, ROW_i and $COLUMN_j$, respectively. $P_{i,j}$ is connected to ROW_i and $COLUMN_j$. The problem of *permutation*

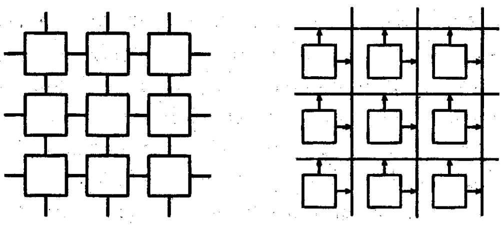

Figure 1: MC and MBUS.

routing on the MBUS is defined as follows: The input is given by n^2 *packets* that are initially held by the n^2 processors, one for each. Each packet, (d, σ), consists of two portions; d is a *destination* address that specifies the processor to which the packet should be moved, and the data portion σ of the packet is an integer. No two packets have the same destination address. Routing requires that all n^2 such packets be moved to their correct destinations.

Our discussion throughout this paper is based on the following three rules on the model: (i) We follow the common practice on how to measure the running time of MBUSs: The one-step computation of each processor P consists of (a) reading the current data on both row and column buses P is connected to, (b) executing arbitrarily complicated instructions using the local memory and (c) if necessary, writing data to the row or column bus, or (possibly different data) to both. The written data will be read in the next step. (ii) The queue size is not bounded, namely, an arbitrary number of packets can stay on a single processor temporarily. (iii) What can be written on the buses by the processor P must be the packet originally given to P as its input packet or one of the packets that have been read so far by P from its row or column bus. (Nothing other than packets can be written.) This means that any kind of data compression is not allowed. (iv) We allow the simultaneous write. However, if two or more packets are written on the same bus simultaneously, then a special value flows, which only shows the collision.

Here is our main result:

Theorem (Main Result). There exists a randomized algorithm for rounting on the two-dementional mesh of buses with high probability in $1.46875n + o(n)$ steps.

3 Randomized Algorithm

In this section we present our randomized algorithm, which moves every packet to its destination using the row bus at most once and the column bus at most once. The algorithm consists of the following two stages: In the first stage, randomly chosen $0.5n^2$ packets first move to their correct places with respect to at least row or column. Then, the remaining $0.5n^2$ packets move to their correct places with respect to at least row or column. In the second stage, all packets move to their final destinations. Here is our algorithm:

Algorithm: *Random Rout*

Stage 1-1. For simplicity, we assume that n can be divided by four. The whole $n \times n$ plane is divided into four $\frac{n}{2} \times \frac{n}{2}$ planes. In this stage, randomly chosen packets on the upper-left $\frac{n}{2} \times \frac{n}{2}$ and on the lower-right $\frac{n}{2} \times \frac{n}{2}$ planes are moved horizontally using row buses. At the same time, randomly chosen packets on the lower-left $\frac{n}{2} \times \frac{n}{2}$ and on the upper-right $\frac{n}{2} \times \frac{n}{2}$ planes are moved vertically using column buses. And the upper-left plane, $\frac{n}{2}$ processors in each row are devided into $0.125n$ horizontal *blocks* each of which consists of 1×4 continuous processors, $(P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4})$, $(P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8})$, $\cdots$, $(P_{1,\frac{n}{2}-3}, P_{1,\frac{n}{2}-2}, P_{1,\frac{n}{2}-1}, P_{1,\frac{n}{2}})$. Randomly chosen two of each four processors write their initial packets on the row bus. One can see that there are $\frac{n}{8}$ blocks on each row in the upper-left plane. Similarly for the lower-right plane. Which processors in the upper-left plane write their initial packets on the row buses is determined by the following two phases, Phase 1 and Phase 2, and the same for the lower-right plane. The upper-right and the lower-left planes are divided into vertical blocks of size 4×1. Randomly chosen two of four processors in each block write their initial packets on the column bus. Which processors write their initial packets on the column buses is very similar to the following: The follwoing two phases are executed in each block, from left to right, the 1st through $\frac{n}{2}$th rows in parallel.

Phase 1. Two processors $P_{i,4j+1}$ and $P_{i,4j+2}$ write their initial packets on the row bus with probability $\frac{1}{2}$.

Phase 2. According to Phase 1, one of the following four operations is executed:

1. If only $P_{i,4j+1}$ wrote the packet, then $P_{i,4j+3}$ writes its initial packet on the row bus. Go to the next block. (see Fig. 2 (1).)
2. If only $P_{i,4j+2}$ wrote the packet, then $P_{i,4j+3}$ writes its initial packet on the row bus. Go to the next block. (see Fig. 2 (2).)
3. If both $P_{i,4j+1}$ and $P_{i,4j+2}$ wrote the packets, then $P_{i,4j+1}$ again writes its packet and then $P_{i,4j+4}$ writes its packet on the row bus. Go to the next block. (see Fig. 2 (3).)
4. If neither $P_{i,4j+1}$ nor $P_{i,4j+2}$ wrote the packets, then $P_{i,4j+2}$ again writes its packet and then $P_{i,4j+4}$ again writes its packet on the row bus. Go to the next block. (see Fig. 2 (4).)

/* Since two of each four packets can move, $\frac{n^2}{2}$ packets have moved so far in total. */

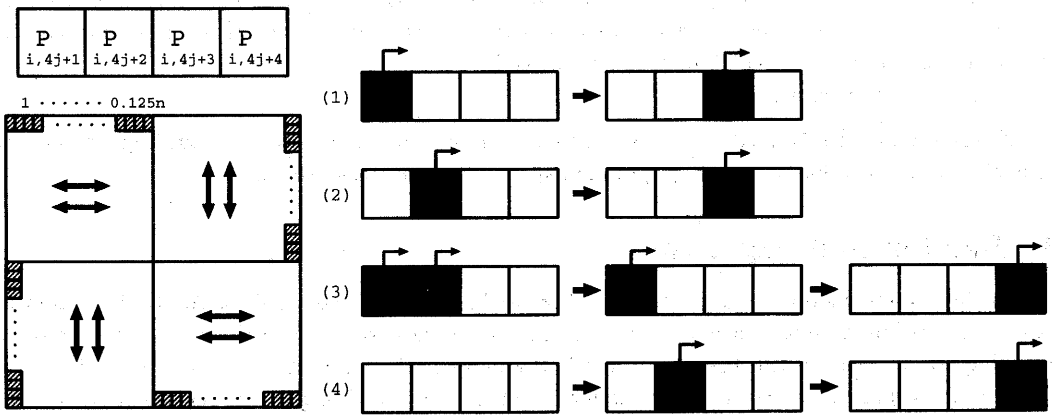

Figure 2: Two packets written on the row bus in Stage 1-1.

Stage 1-2. The remaining $0.5n^2$ packets are moved. However, the directions are switched, i.e., the packets on the upper-right plane are moved vertically using the column buses and so on. Now, the upper-left plane is further divided into 2×1 processors and the same for the lower-right plane. The lower-left and the upper-right planes are divided into 1×2 processors.

As for the upper-left plane, the following two phases are iterated from the top to the bottom blocks of 2×1 processors in parallel for the 1st through $\frac{n}{2}$th columns.

Phase 1. If two processors $P_{2i+1,j}$ and $P_{2i+2,j}$ remain holding their initial packets, then they write the packets on the column bus.

Phase 2. According to Phase 1, one of the following four operations is ececuted:

1. If only $P_{2i+1,j}$ wrote the packet, then go to the next block. (see Fig. 3 (1).)
2. If only $P_{2i+2,j}$ wrote the packet, then go to the next block. (see Fig. 3 (2).)
3. If both $P_{2i+1,j}$ and $P_{2i+2,j}$ wrote the packets, then $P_{2i+1,j}$ writes its packet and then $P_{2i+2,j}$ writes its packet on the column bus. Go to the next block. (see Fig. 3 (4).)
4. If neither $P_{2i+1,j}$ nor $P_{2i+2,j}$ wrote the packet, then go to the next block. (see Fig. 3 (3).)

/* Since the remaining packets can move, $\frac{n^2}{2}$ packets have moved so far in total. */

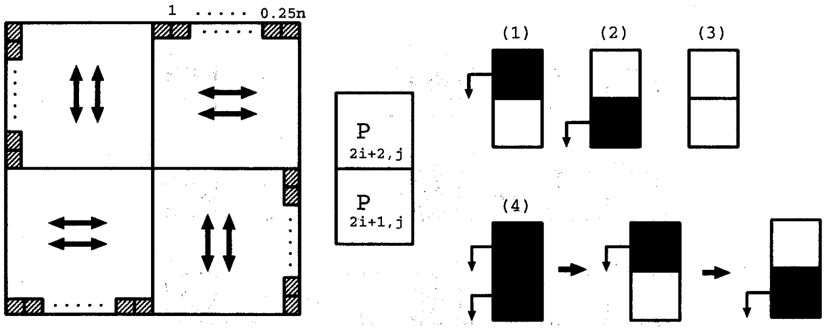

Figure 3: The remaining packets written on the column bus in Stage 1-2.

Stage 2. All the packets move to their final destination. Intuitively, the order of processors writing packets is determined not by the positions of the processors as before but by the destinations of packets. Now, for each row (column), we make $0.5n$ blocks of two neibour processors.

Phase 1 Each processor on each row executes one of the following operations; in the first iteration, packets whose destinations are the leftmost two columns can move, and in the second iteration, pacekts whose destinations are the third and fourth columns can move and so on. Similarly for each colunm. Let L_i and R_i be the packets that destine for the left and the right processors in the ith iteration:

1. If a processor holds either L_i or R_i, then the packet is written on the row bus.
2. If a processor holds at least three packets, L_i, R_i and another, then the packet other than L_i and R_i is written.
3. If a processor holds L_i and R_i but does not hold any other packets, then L_i is written.
4. If a processor holds neither L_i nor R_i, then the processor sleeps.

In the ith iteration, there are the following five cases on which packets are riding on the row bus: (i) L_i is written. Then we let $\mathit{flow}_i = 1$. (ii) R_i is written. Then we let $\mathit{flow}_i = 2$. (iii) No packets are written. Then we let the value $\mathit{flow}_i = 3$. (iv) A packet other than L_i and R_i is written. Then we let $\mathit{flow}_i = 4$. (v) The collision happens. Then we let $\mathit{flow}_i = 5$.

Phase 2 One of the following is executed:

1. If flow_i is 1, 2 or 3, then go to the next block. (see Fig. 4 (1), (2), (3).)
2. If $\mathit{flow}_i = 4$, then L_i is moved first and then R_i is moved. (see Fig. 4 (4).)

3. In the case of $flow_i = 5$, there are further two cases according to $flow_{i-1}$:
$flow_{i-1} \neq 1$. L_i is moved first and then R_i is moved. (see Fig. 4 (5).)
/* In this case, since all processors can recognize that the collision comes from L_i and R_i, L_i and R_i can be moved in the next two steps. */
$flow_{i-1} = 1$. (i) If a processor holds R_{i-1}, then it is written on the row bus in this time-slot; otherwise the processor sleeps (see Fig. 4 (6)). Then the ith iteration is executed again. (ii) If no packet is written (i.e., no processor holds R_{i-1}), then L_i is moved and then R_i is moved.
/* In the case of $flow_{i-1} = 1$, R_{i-1} may remain unwritten. Namely, one cannot see which packets cause the collision since three packets L_i, R_i and R_{i-1} can be written. Thus we need one extra time-slot for R_{i-1}. */

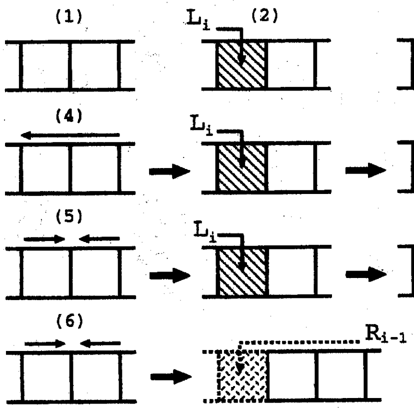

Figure 4: Some packets reachs their destination in Stage2.

4 Analysis (Proof of Theorem)

The argorithm shown in section 3 takes different steps on each row and column bus in each stage. Indeed, each stage is terminated after certain steps even if all packets are not processed. In this section, we first calculate that the expected number of steps for Stage 1-1, Stage 1-2 and Stage 2 are $0.3125n$, $0.375n$ and $0.78125n$ steps, respectively. Finaly, we prove that at most $1.46875n + o(n)$ steps are sufficient to move all the packets to their final positions, with high probability.

Stage 1-1. Consider a block of four processors $P_{i,4j+1}$, $P_{i,4j+2}$, $P_{i,4j+3}$ and $P_{i,4j+4}$. There are the four cases according to which processors write on the bus. (See Stage 1-1 in Section 3 again.) Since the each $P_{i,4j+1}$ and $P_{i,4j+2}$ writes on the row bus with probability $\frac{1}{2}$, each case in Phase 2 happens with probability $\frac{1}{4}$. Recall that we need two steps for each block if case 1 or 2 happens, and three steps if case 3 or 4 happens. On average, we need 2.5 steps for each block. Hence, the expected number of steps for all $0.125n$ blocks is $0.3125n$.

Stage 1-2. In this stage, we consider a block of two processors $P_{2i+1,j}$ and $P_{2i+2,j}$. There are the four cases according to whether each of two processors has already written its initial packet in Stage 1-1. (See Stage 1-2 in section 3.) The probability that each packet remains unwritten after Stage 1-1 is $\frac{1}{2}$. Hence, each of the four cases in Phase 2 happens equally with probability $\frac{1}{4}$. Each block of processors needs one step if case 1, 2 or 3 happens, and three steps if case 4 happens. Therefore, 1.5 steps for each block are needed steps in average. Thus, the expected number of steps needed in this stage for all $0.25n$ blocks is $0.375n$.

Stage 2. After Stage 1-1 and Stage 1-2, every packet arrives at its correct place with respect to at least row or column. Consider an arbitrary packet x. The probability that x is in its correct place with respect to row or column is equally $\frac{1}{2}$. Suppose, for example, that x is in its correct place with respect to row. Then, it uses the column bus to move to its destination in this stage. Namely, x uses each of row and column bus with probability $\frac{1}{2}$.

Recall that two neighboring processors are considered to be a block for each column (row). In this section, we only consider the block of two processors on the same row. (The same argument holds for the block on column.) Suppose that L_i and R_i be packets whose destinations are P_{2i-1} and P_{2i}, respectively. Here, we assume that there is no processor that has both L_i and R_i. (We will consider the other case later.) Consider the row bus B that is connected to P_{2i-1} and P_{2i}. Then, one of the following four cases happens with probability $\frac{1}{4}$.

1. Only L_i is on B, i.e., only L_i uses B to move to P_1.
2. Only R_i is on B, i.e., only R_i uses B to move to P_2.
3. Neither L_i nor R_i is on B, i.e., B is not used by L_i and R_i.
4. Both L_i and R_i are on B, i.e., the collision happens.

It takes one step for each block in the cases 1 through 3. In the case 4, if the state of the previous block is 1, i.e., $flow_{i-1} = 1$, it takes four steps, and if the state of the previous block is 2 through 4, i.e., $flow_{i-1} \neq 1$, it takes three steps. Therefore, each block needs $\frac{25}{16}$ steps, and hence the expected number of steps for $0.5n$ processors is $0.78125n$.

Next, we consider the other case in which there exists some prosessor P which has both packets L_i and R_i. In this case, for each block, the probability that the case 4 happens decreases, and the probability that the case 1 happens increases. However, the number of steps needed for each block is less than $\frac{25}{16}$ in average by the following reason: Note that since P has two packets, it first writes L_i and then writes R_i on the bus B. There are two cases: (i) There exists a processor on this row that has a packet L_{i+1} or R_{i+1} whose destination is one of the processors of the next block. (ii) There is no such processor on this row. In each case, L_i can be moved without collision. However, when R_i is moved, there happens a collision in the case (i). Thus, it takes three steps and two steps in the cases (i) and (ii), respectively. Hence, the expected number of steps needed for each block is less than $\frac{25}{16}$.

As a result, the total expected number of steps for Stage1-1, Stage1-2 and Stage2 is $1.46875n$. Now, we shall prove by applying Chernoff Bounds [1] that if n is enough large value, every packets indeed reachs its destination in those steps with high probability.

Lemma. (Chernoff Bounds [1]) Let $X_1, X_2, ..., X_n$ be independent Bernoulli trials with $\Pr(X_i{=}1) = p$, $\Pr(X_i{=}0) = 1 - p$, $0 < p < 1$. Let $X = \sum_{i=1}^{n} X_i$. Then, for any $0 < \varepsilon < 1$,

$$Pr\,[\, X > (1+\varepsilon)np \,] < \exp\,(\frac{-\varepsilon^2 np}{3})$$

Proof of Theorem. Let $X_{1i} = 1$ when three steps are needed and $X_{1i} = 0$ when two steps are needed for the ith block from the left in Stage 1-1. Then, X_1 have a binomial distribution $B(0.125n, \frac{1}{2})$. Apply the Chernoff bounds with $\varepsilon = \frac{c_1}{0.0625n}\sqrt{n \ln n}$ for $c_1 > 0$.

$$\Pr\,[\, X_1 > 0.0625n + c_1\sqrt{n \ln n}\,] < \exp\,(-\frac{512c_1^2}{3}\ln n) = n^{-d_1} \qquad d_1 > 0$$

Namely, the number of blocks that needs three steps is less than $0.0625n + c_1\sqrt{n \ln n}$. Also, the number of blocks that needs two steps is $0.125n(1-p) - c_1\sqrt{n \ln n}$. We can estimate the following number of steps needed in Stage 1-1.

$$3 \times (0.0625n + c_1\sqrt{n\ln n}) + 2 \times (0.0625n - c_1\sqrt{n\ln n}) = 0.3125n + c_1\sqrt{n\ln n}$$

Therefore, with probability at least $1-n^{-d_1}$, $0.5n^2$ packets is written on the bus in $0.3125n+c_1\sqrt{n\ln n}$ steps. The number of steps needed in Stage 1-2 is obtained as well as Stage 1-1.

$$\Pr [\ X_2 < 0.0625n + c_2\sqrt{n\ln n}\ ] < n^{-d_2} \qquad d_2 > 0$$

$$3 \times (0.0625n + c_2\sqrt{n\ln n}) + 1 \times (0.1875n - c_2\sqrt{n\ln n}) = 0.375n + 2c_2\sqrt{n\ln n}$$

Therefore, with probability at least $1 - n^{-d_2}$, the remaining $0.5n^2$ packets is written on the bus in $0.375n + 2c_2\sqrt{n\ln n}$ steps.

In Stage 2, let $X_{3i} = 1$ when three or more steps are needed and $X_{3i} = 0$ when one step is needed for the ith block from the left.

$$\Pr [\ X_3 < 0.125n + c_3\sqrt{n\ln n}\ ] < n^{-d_3} \qquad d_3 > 0$$

At the moment, the number of blocks needed one step is $0.375n - c_3\sqrt{n\ln n}$. Next, let $k = 0.125n + c_3\sqrt{n\ln n}$, and let $X_{4i} = 1$ when four steps are needed for k blocks and $X_{4i} = 0$ when three steps for k blocks. Apply the Chernoff bounds with $\varepsilon = \frac{c_4}{kp}\sqrt{k\ln n}$ for $c_4 > 0$.

$$\Pr [\ X_4 < kp + c_4\sqrt{k\ln n}\ ] < n^{-d_4} \qquad d_4 > 0$$

$$4 \times (kp + c_4\sqrt{k\ln n}) + 3 \times (k(1-p) - c_4\sqrt{k\ln n}) = 0.40625n + \frac{13}{4}c_3\sqrt{n\ln n} + c_4\sqrt{k\ln n}$$

Since the number of blocks needed one step is $0.375n - c_3\sqrt{n\ln n}$ with probability at least $(1-n^{-d_3})$, in Stage2, with probability at least $(1 - n^{-d_3})(1 - n^{-d_4})$, all packets reaches its destination in $0.78125n + \frac{9}{4}c_3\sqrt{n\ln n} + c_4\sqrt{k\ln n}$ steps.

We have n rows and n columns. So, the probability of bad behavior in at least one row or column can become as large as n times. However, since all the probabilities above the term of n^{-d} for an arbitrary large constat d, this does not cause any serious problem. Therefore, the whole algorithm takes at most $1.46875n + o(n)$ steps with high probability. □

References

[1] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Mathematical Statistics*, vol.23 (1952) 493-507.

[2] S. Cheung and F.C.M. Lau, "A lower bound for permutation routing on two-dimensional bused meshes," *Information Processing Letters*, 45 (1993) 225-228.

[3] K. Iwama, E. Miyano, and Y. Kambayashi, "Routing Problems on the Mesh of Buses," *J. Algorithms*, 20 (1996) 613-631.

[4] K. Iwama and E. Miyano, "Oblivious routing algorithms on the mesh of buses," *Proc. IPPS*, (1997) 721-727.

[5] L.Y.T. Leung and S.M. Shende, "On multidimensional packet routing for meshes with buses," *J. Parallel and Distributed Computing*, 20 (1994) 187-197.

[6] M. Tompa, "Lecture notes on message routing in parallel machines," Technical Report # 94-06-05, Department of Computer Science and Engineering, University of Washington, 1994.