Multinode broadcasting in a wormhole-routed 2-D torus using an aggregation-then-distribution strategy

Y.-S.Chen and C.-Y.Chen

Abstract: An efficient multinode broadcasting algorithm in a wormhole-routed 2-D torus is presented where there are an unknown number of s source nodes located on unknown positions each intending to broadcast a message of size m bytes to the rest of network. The torus is assumed to use the all-port model and the popular dimension-ordered routing. Most existing results are derived based on finding multiple edge-disjoint spanning trees in the network. The main technique used is an aggregation-then-distribution strategy. First, the broadcast messages are aggregated into some positions of the torus. Then, a number of independent subnetworks are constructed from the torus. These subnetworks, which are responsible for distributing the messages, can well exploit the communication parallelism and the characteristic of wormhole routing. It is shown that such an approach is more appropriate than those using edge-disjoint trees for fixed-connection network such as tori. This is justified by performance analysis.

1 Introduction

A massively parallel computer (MPC) consists of a large number of identical processing elements interconnected by a network. One basic communication operation in such a machine is broadcasting. Two commonly discussed instances are one-to-all broadcast and all-to-all broadcast, where one or all nodes need to broadcast messages to the rest of the nodes. A more complicated instance is the many-to-all (or multinode) broadcast, where an unknown number of nodes located in unknown positions each intending to perform a broadcast operation. The focus of this paper is on the multinode broadcast problem, whose many applications can be found in parallel graph algorithms, parallel matrix algorithms, fast Fourier transformation, parallel compilation, and cache coherence. In addition to multinode broadcasting, many collective communication patterns, such as one-to-all broadcasting, all-to-all broadcasting, complete exchange, scatter, gather, and reduction, have all intensive attention recently in [1, 2].

The multinode broadcast problem has been studied on a variety of interconnection networks [3–8]. Saad and Schultz [3, 4] initially defined this problem and proposed a simple routing algorithm for hypercubes. Stamoulis and Tsitsiklis [5] proposed to use n edge-disjoint spanning trees in an n-cube to solve this problem. A similar related work is to embed a complete binary tree in the star graph, proposed by Tseng and Chen *et al.* [9]. A distributed approach to improve the load imbalance problem in [5] was presented by Tseng [6] for hypercubes and star graphs. Efforts were made by Varvarigos [7] to solve the more complicated problem where each source node may have several messages (of the same length) to broadcast. More recently, Susanne *et al.* [8] proposed a scheme called s-to-p broadcasting, where the authors tried to align the broadcast messages into a regular pattern before they are distributed.

The aforementioned results are all based on finding edge-disjoint spanning trees in a network and are appropriate for nonfixed connection networks [10, 11]. One problem with this is that the number of edge-disjoint trees that could be offered by a network is fixed [10]. The other problem is that the characteristic of wormhole routing, which is assumed in this paper, is not well exploited in [12]. We study the scheduling of message distribution for many-to-all broadcast in a wormhole-routed 2-D torus, which type of architecture has been adopted by parallel machines such as Cray T3D and T3E (3-D tori). Observe that 2-D and 3-D torus are fixed-connection networks. The recently popular wormhole routing technology is assumed. Sending a packet involves two costs: *start-up-time* and transmission time. Attempts to minimise both these costs are made.

Our approach is based on an aggregation-then-distribution strategy. First, the network partitioning techniques proposed in [12] are used to get multiple independent subnetworks (which are different from edge-disjoint spanning trees) in a torus. The number of independent subnetworks is actually an adjustable parameter. Given a multinode broadcast problem with an unknown number of s source nodes located on unknown positions in an $n \times n$ torus each intending to broadcast an m-byte message, our approach can solve it efficiently in time $O(\lceil \log_5 n \rceil T_s + \max\{\lceil \log_5 \lceil n/h \rceil, h\} \rceil (s/h) m T_c)$, where h is the number of independent subnetworks. It is shown that this number has outperformed the aforementioned schemes using edge-disjoint-spanning trees.

2 Basic idea

2.1 System model

A massively parallel computer is formally represented as $G = (V, C)$, where V denotes the node set and C specifies

IEE Proceedings online no. 20000887

DOI: 10.1049/ip-cdt:20000887

Paper first received 20th July 1999 and in revised form 22nd September 2000

The authors are with the Department of Statistics, National Taipei University, Taipei, 10433, Taiwan

the channel connectivity. Each node contains a separate router to handle its communication tasks. We consider G as a two-dimensional torus $T_{n_1 \times n_2}$ with $n_1 \times n_2$ nodes. Each node is denoted as $P_{i,j}$, $1 \le i \le n_1$, $1 \le j \le n_2$ and P_{i_1,i_2} has an edge connected $P_{(i_1 \pm 1) \bmod n_1, i_2}$ along dimension one and an edge to $P_{i_1,(i_2 \pm 1) \bmod n_2}$ along dimension two. Each edge is considered consisting of two directed communication links pointing in opposite directions.

The wormhole routing model is assumed [13]. Under such a model, each packet is partitioned into smaller units called flits, which are sent in a pipelined manner. In the absence of congestion, the communication latency in the networks is proportional to the sum of message length and routing distance. Specifically, the time required to deliver a packet of L bytes from a source node to a destination node can be formulated as $T_s + LT_c$, where T_s is the start-up time containing the channel setup and software overhead, and T_c represents the transmission time per data byte. In this paper, attempts are made to counter the trade-off between the start-up and the transmission costs.

In addition we adopt the all-port model, in that a node can simultaneously send and receive messages along all outgoing and incoming links, and the dimension-ordered routing [6], in that every message must travel in a strictly increasing order in terms of link dimensions.

2.2 Network partitioning

Our work is based on partitioning the torus into some subnetworks. In the following we review some definitions, based on the work in [11]. Consider a torus $T_{n_1 \times n_2}$. Suppose h is an integer which divides both n_1 and n_2. We define k *data-distribution network* $DDN_k = (V_k, C_k)$, $k = 0..h-1$ as follows:

$$V_k = \{p_{i,j} | i = ah + k, j = bh + k, \text{ for all } a = 0..(n_1/h) - 1 \text{ and } b = 0..(n_2/h) - 1\}$$

$$C_k = \{\text{all channels at rows } ah + k \text{ and column } bh + k\}$$

Intuitively, each DDN is a dilation-h torus of size $(n_1/h) \times (n_2/h)$, in the sense that each edge is dilated by a path of h edges. An example is shown in Fig. 1 with four dilated-4, 4×4 tori embedded in a 16×16 torus. Also, we partition the $T_{n_1 \times n_2}$ into $n_1 \times n_2/h^2$ data collecting network

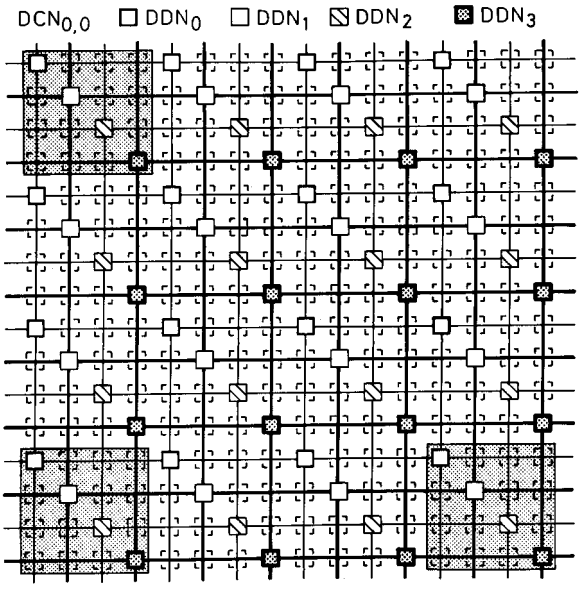

Fig. 1 *Network partioning scheme*

$DCN_{a,b} = (V_{a,b}, C_{a,b})$, $a = (0..n_1 - 1)/h$, $b = (0..n_2 - 1)/h$, as follows:

$$V_{a,b} = \{p_{i,j} | i = a \times h + x, j = b \times h + y \text{ for all } x, y = 0..h - 1\}$$

$$C_{a,b} = \{\text{the set of edges induced by } V_{a,b} \text{ in } T_{n_1 \times n_2}\}$$

Intuitively, these DDNs are obtained by evenly slicing the torus into $n_1 \times n_2/h^2$ blocks, each being a square $h \times h$ submesh. Fig. 1 illustrates this definition when $h = 4$. According to [12], these DDNs and DCNs have the following properties:

- $DDN_0, DDN_1, \ldots, DDN_{h-1}$ are mutually independent (under the given port model)
- $DCN_0, DCN_1, \ldots, DCN_{k-1}$ are mutually independent (under the given port model) and they together contain all nodes of G.
- DDN_i and DCN_j intersect in at least one node, for all $0 \le i < h$ and $0 \le j < (n_1 \times n_2/h^2)$.
- $DDN_0, DDN_1, \ldots, DDN_{h-1}$ are isomorphic.
- $DCN_0, DCN_1, \ldots, DCN_{k-1}$ are isomorphic.

2.3 Aggregation-then-distribution strategy

Existing multinode broadcasting schemes [5–7] mainly use multiple spanning trees such that all source nodes are evenly distributed submessages to root nodes of all spanning trees. As mentioned these approaches are suitable for the nonfixed connection network [5–7]. Observe that exploiting the maximum number of multiple spanning trees is not feasible for the fixed-connection network [12, 14]. For instance, the maximum number of spanning trees in 2-D tori is four. In 2-D tori, s source nodes evenly deliver their messages to four disjoint spanning trees. The limitation in number of spanning trees can be efficiently improved by our *aggregation-then-distribution scheme*, because our scheme utilises the property of the network partitioning scheme.

The original 2-D network is partitioned into arbitrary size of h DDNs and k DCNs. The main function of aggregation-then-distribution scheme is outlined as follows.

2.3.1 Aggregation phase: The broadcast messages are aggregated into some regular positions of the torus. That is, the aggregation phase is to regularise the data pattern from unknown location of s source nodes. Observe that the load imbalance problem unfortunately occurs. A tuning operation is needed to overcome the load imbalance problem.

2.3.2 Distribution phase: A number of independent subnetworks are constructed. These subnetworks, which are responsible for distributing the messages, can well exploit the communication parallelism and the characteristic of wormhole routing. The distribution phase is to broadcast a message on every DDN in parallel.

3 Multinode broadcasting scheme

3.1 Aggregation phase

Consider a torus $T_{n \times n}$ with n^2 nodes; s source nodes intend to broadcast to the rest of the network. Messages of source nodes are initially aggregated into some diagonals of partitioned subtori DCNs. This operation is very efficient

for the many-to-all collective communication problem as the communication pattern has been regularised in advance. The aggregation scheme is divided into two steps: diagonal-based data-aggregation operation, and balancing-load operation.

Section 3.1.1 presents a fundamental operation, namely a diagonal-based data-aggregation operation. Susanne *et al.* proposed s-to-p routing [8] adopting similar location-repositioning work. The main difference is that our algorithm is to aggregate messages into disjoint subtori DDN_0, $DDN_1, \ldots,$ and DDN_{h-1}. Unfortunately, diagonal-based data aggregation causes a load imbalance problem. To overcome this, a balancing-load operation is performed in Section 3.1.2. Our approach exploits the communication parallelism rather than the s-to-p routing.

3.1.1 Diagonal-based data-aggregation operation:

The sizes of DDN_0, $DDN_1, \ldots, DDN_{h-1}$ and DCN_0, $DCN_1, \ldots, DCN_{k-1}$ are initially determined. It is a trade-off problem to determine the value of h and k. Basically, the higher the value of h is, the lower latency will be.

Given a node $P_{i,j}$ and an integer k, let $D(P_{i,j}, k)$ denote a sequence of k nodes. For instance, the main diagonal in a square $T_{n\times n}$ torus passing node $P_{0,0}$ is the sequence $D(P_{0,0}n)$. The node $P_{i,j} = P_{0,0}$ then the sequence of the diagonal nodes is $P_{1,1}, P_{2,2}, \ldots,$ and $P_{n-1,n-1}$. A torus can be viewed as n diagonals $D(P_{i,0}, n)$ or L_i, $i = 0..n-1$. The purpose of the data-aggregation operation is to aggregate n diagonals into $\tilde{L}_j$ diagonals, where $j = 0..\lceil n/h \rceil - 1$ and $\tilde{L}_j = L_{j'}$ for $j' = j^*\lceil n/h \rceil$. In other words, data are aggregated into the main diagonal for every *DCN*.

The time-cost of data aggregation depends on the value of h. During each data-aggregation operation, every node $P_{i,j}$ in the main diagonal of each *DCN* has aggregated messages from nodes $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-2}$, and $P_{i,j+2}$ as shown in Fig. 2*a*. This also implies that $\tilde{L}_j$ or $L_{j'}$ diagonals aggregates messages from $L_{-2+j'}$, $L_{-1+j'}$, $L_{1+j'}$ and $L_{2+j'}$ as illustrated in Fig. 2*b*. All communication patterns are clearly congestion-free. The communication latency is determined by the size of h, not the value of n. If $h > 4$, the data-aggregation operation can be recursively executed within $\lceil \log_5 h \rceil$ time units. Fig. 2*c* illustrates this operation if $h = 25$. Therefore we have the following result if $h > 4$:

Lemma 1: Diagonal-based data-aggregation operation is recursively performed on a $T_{n\times n}$ within time

$$\lceil \log_5 h \rceil T_s + \sum_{i=1}^{\lceil \log_5 h \rceil - 1} 5^{i-1} m T_c = \lceil \log_5 h \rceil T_s + \frac{5^{\lceil \log_5 h \rceil} - 1}{4} m T_c$$

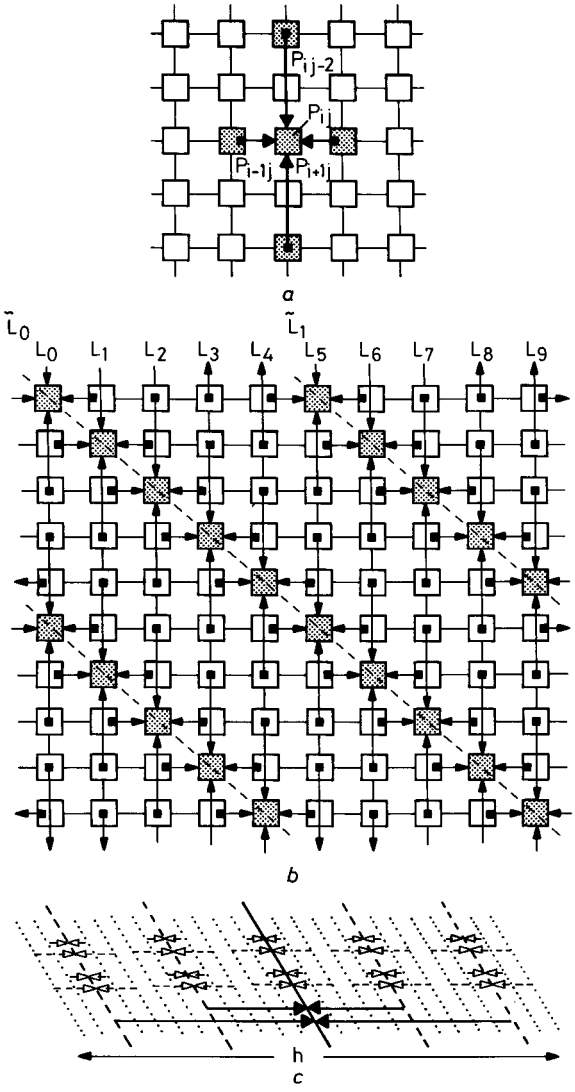

Fig. 2 *Data aggregation*

a Aggregation pattern
b Aggregation operation $h = 5$
c Aggregation operation when $h = 25$

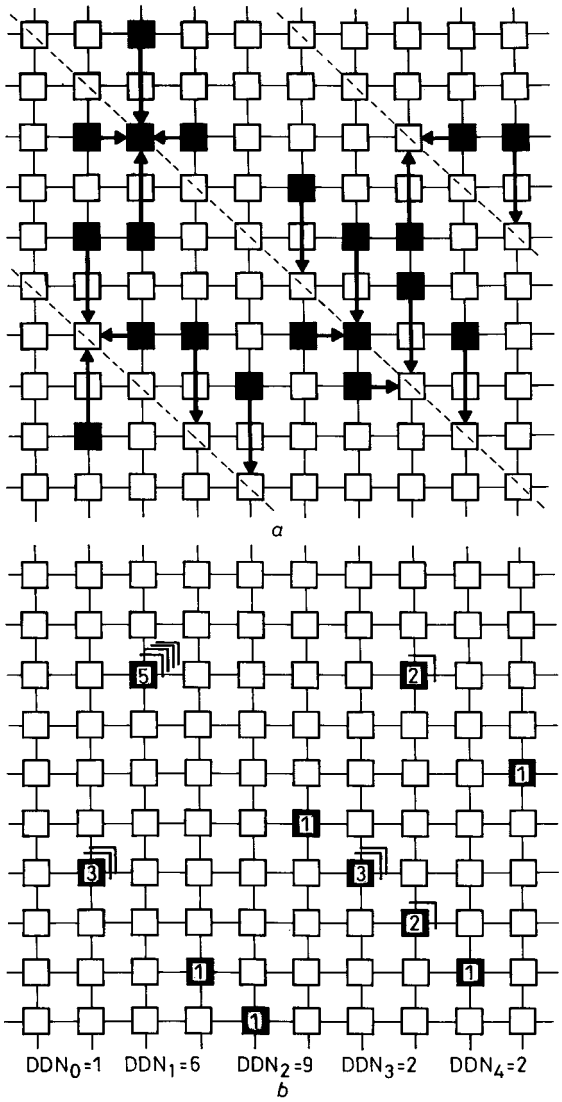

Fig. 3 *Data aggregation*

a Diagonal based
b Aggregation result

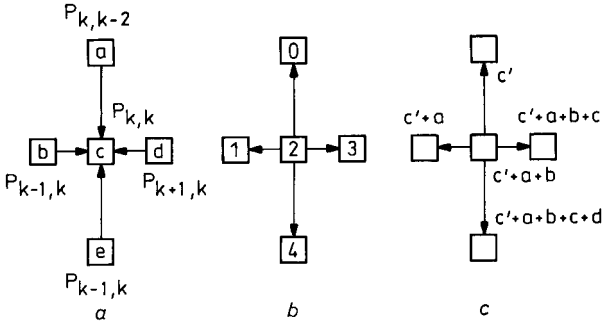

Fig. 4 *Simple prefix-sum procedure for five nodes*

a Basic forwarding stage
b Order of backward
c Basic backward stage

where $h > 4$.

Further, each data-aggregation operation may aggregate messages from partial nodes by four neighbouring nodes if it is one of the source nodes. Consider 20 source nodes intending to send a message to the rest of the network as shown in Fig. 3*a*. After performing data-aggregation once, the result is shown in Fig. 3*b*. Assume that $h = 5$, the work loads, of DDN_0, DDN_1, DDN_2, DDN_3, and DDN_4 are 1, 6, 9, 2, and 2. Obviously, there is a load imbalance.

3.1.2 Balancing-load operation: Recall the aggregation result in Fig. 3*b*; every one of *DDN*s has different amount of messages. A balancing-load operation aims to achieve the load balance. This operation is achieved by maintaining a prefix-sum value. Using the prefix-sum value allows us to execute a data tuning procedure. Finally, every *DDN* will keep the same amount of messages. This operation makes our algorithm with high parallelism. Two main procedures are

- **prefix-sum procedure**: calculate prefix-sum value
- **data tuning procedure**: perform message tuning procedure to achieve the load balance

3.1.2.1 Prefix-sum procedure: First, the prefix-sum procedure is to exchange information of the amount of collected messages to get the prefix-sum value. This procedure only propagates the control message across the 2-D tori network. The prefix-sum procedure needs (*a*) forward stage and (*a*) backward stage. For ease of presentation a simple prefix-sum procedure for five nodes is initially explained.

(i) **Basic forward stage**: Node $P_{k,k}$ containing message c receives messages a, b, d, and e from nodes $P_{k,k-2}$, $P_{k-1,k}$, $P_{k+1,k}$, and $P_{k,k+2}$, as illustrated in Fig. 4*a*. Note that node $P_{k,k}$ must maintain values of a, b, c, d, and e on each forward stage to calculate a partial prefix-sum in future backward stage.

(ii) **Basic backward stage**: Assume that node $P_{k,k}$ gets a local partial prefix-sum value c' (from the previous backward stage), then node $P_{k,k}$ must send backward value c' plus partial prefix-sum to nodes $P_{k,k-2}$, $P_{k-1,k}$, $P_{k+1,k}$, and $P_{k,k+2}$, according to the order shown in Fig. 4*b*. That is, node $P_{k,k}$ sends value of c', $c'+a$, $c'+a+b$, $c'+a+b+c$, and $c'+a+b+c+d$ to nodes $P_{k,k-2}$, $P_{k-1,k}$, $P_{k,k}$, $P_{k+1,k}$, and $P_{k,k+2}$, respectively, as illustrated

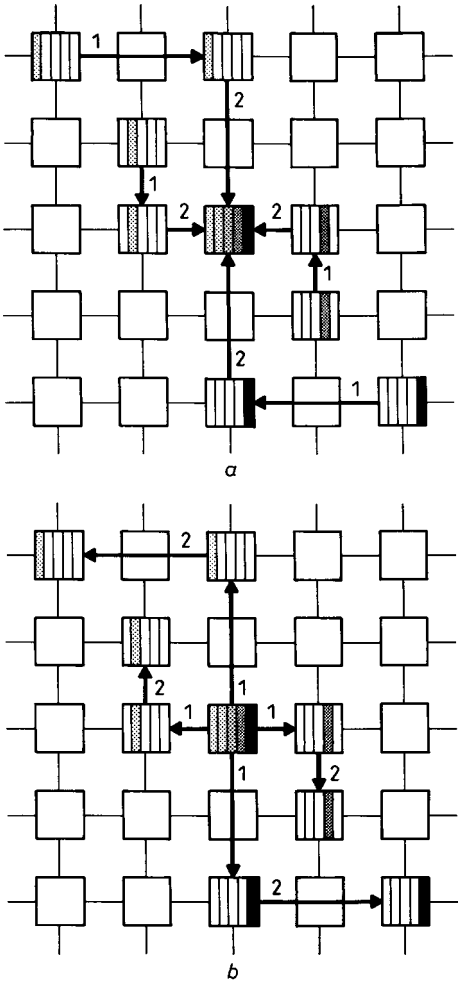

Fig. 5 *Simple prefix-sum procedure for five nodes*

a First forward
b Second forward
c First backward
d Second backward

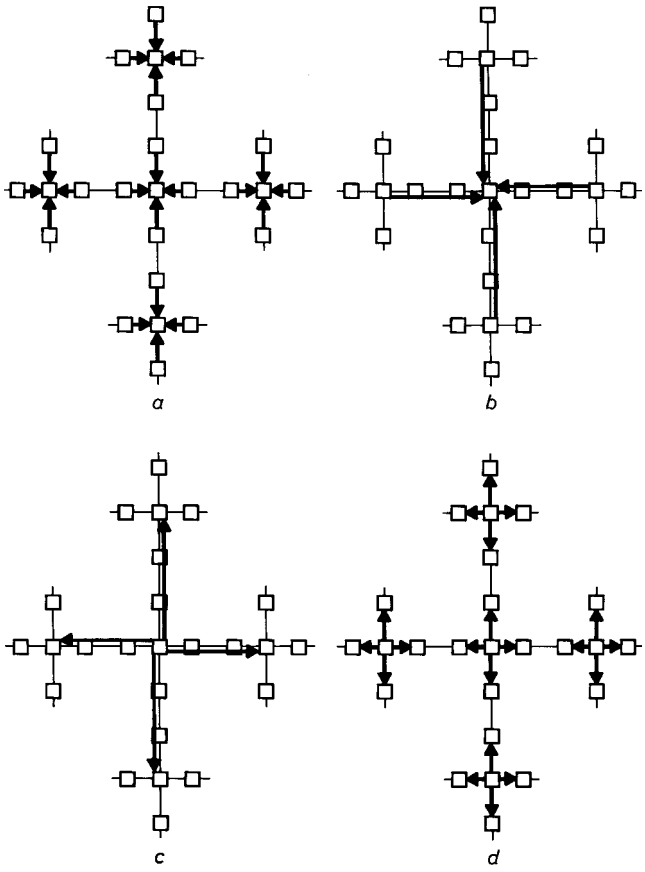

Fig. 6 *Simple prefix-sum procedure for five nodes*

a Basic forward on diagonal
b Basic backward on diagonal

in Fig. 4*c*.

Due to the fact that all nodes with messages are always located in the diagonal of tori (by the data-aggregation operation), so a diagonal-based recursive prefix-sum procedure is needed. Nodes $P_{k-2,k-2}$, $P_{k-1,k-1}$, $P_{k+1,k+1}$, and $P_{k-2,k+2}$ are located in a diagonal, so two communication steps are performed as illustrated in Fig. 5. The first communication step is to let diagonal nodes $P_{k-2,k-2}$, $P_{k-1,k-1}$, $P_{k+1,k+1}$ and $P_{k+2,k+2}$ send corresponding messages to $P_{k,k-2}$, $P_{k-1,k}$, $P_{k+1,k}$, and $P_{k,k+2}$ as shown in Fig. 4*a*. Clearly, this communication step is congestion-free and takes one time step. A second communication step performs the basic forward stage. Similarly, for the backward stage in reverse. Collectively, each of the diagonal-based forward and backward stages takes two time steps. Now recursively perform the basic forward and backward stages for $\lceil \log_5 n \rceil$ times, as shown in Fig. 6. Each of the recursive forward and backward stage works within time $\lceil \log_5 n \rceil (T_s + T_c)$. This implies that the recursive forward and backward stage on the diagonal needs time $2\lceil \log_5 n \rceil (T_s + T_c)$. Further, incoming data must be kept for the backward stage, so each node must use $5\lceil \log_5 n \rceil$ extra memory. Consequently, the time complexity of prefix-sum procedure is $4\lceil \log_5 n \rceil (T_s + T_c)$. For example, a prefix-sum procedure is operated in Figs. 7–9. First, all information is collected into the main diagonal as shown in Fig. 7*a* (this operation is also a basic forward stage). The

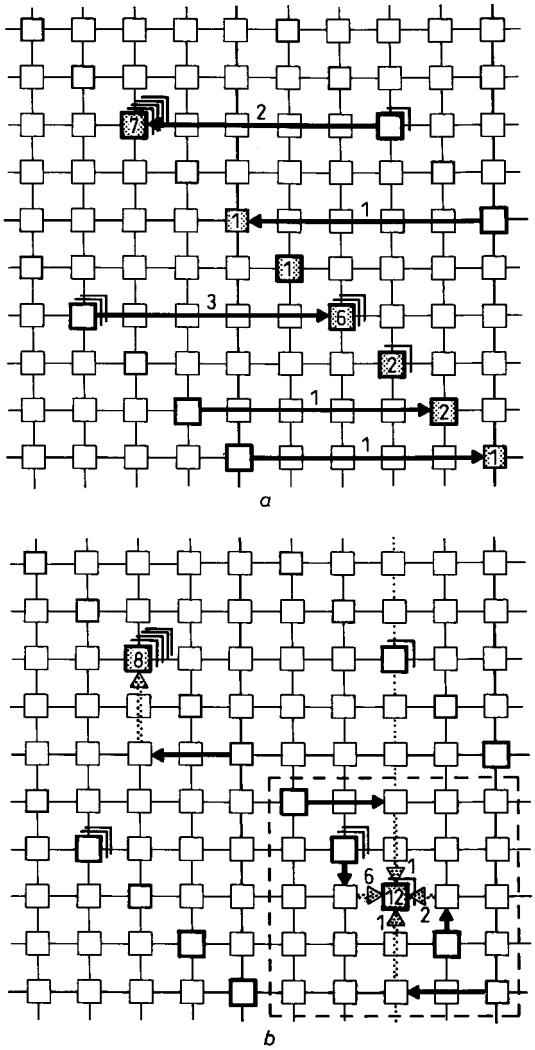

Fig. 7 *Example of prefix-sum procedure*

a Collecting information into main diagonal
b First forward stage

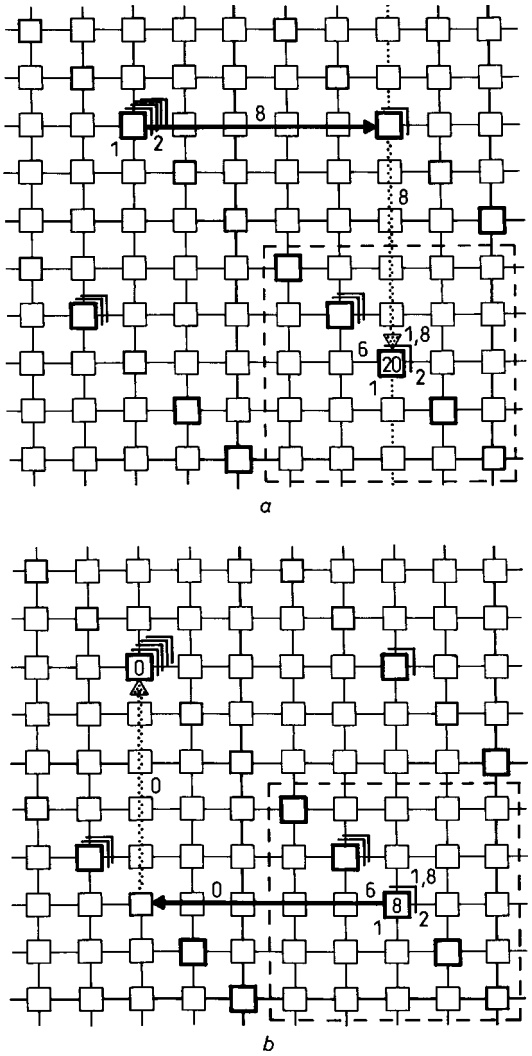

Fig. 8 *Example of prefix-sum procedure*

a Second forward stage
b First backward stage

first and second forward stages on the diagonal are shown in Figs. 7*b* and 8*a*. Then the first and second backward stages on the diagonal are displayed in Figs. 8*b* and 9*a*. Finally, all partial prefix-sums are distributed from the main diagonal (this operation is a basic backward stage) and then the final result of the prefix-sum procedure is obtained in Fig. 9*b*.

3.1.2.2 Data tuning procedure: The data tuning procedure balances the work load among all *DDN*s. The task is split into two parts: finding a destination list, and performing data-movement operation. For each node with broadcast messages, two important pieces of information is kept: the prefix-sum value and the number of broadcast messages. Using these two values, a destination list can be obtained. For node x, which is located in DDN_j, each element i in the destination list indicates that node x should move one copy of message to DDN_i. The detailed algorithm is given below.

(i) **Finding a destination list**: The prefix-sum value and number of messages are α and β respectively. The original destination list is $\{\alpha, \alpha+1, \ldots, \alpha+\beta+1\}$. If the number of *DDNs* is h, let the destination list be $F = \{\alpha \bmod h, (\alpha+1) \bmod h, \ldots, (\alpha+\beta+1) \bmod h\}$. For instance in Fig. 10, if $h = 5$ one node whose prefix sum is 9, the number of collected messages is 3 and the original destination list is

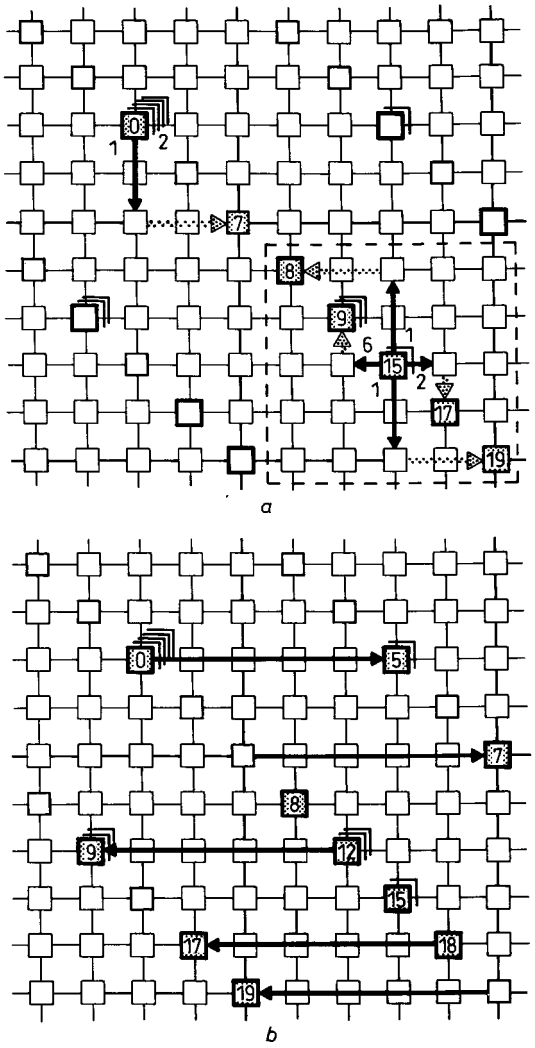

Fig. 9 *Example of prefix-sum procedure*
a Second backward stage
b Distribution from main diagonal

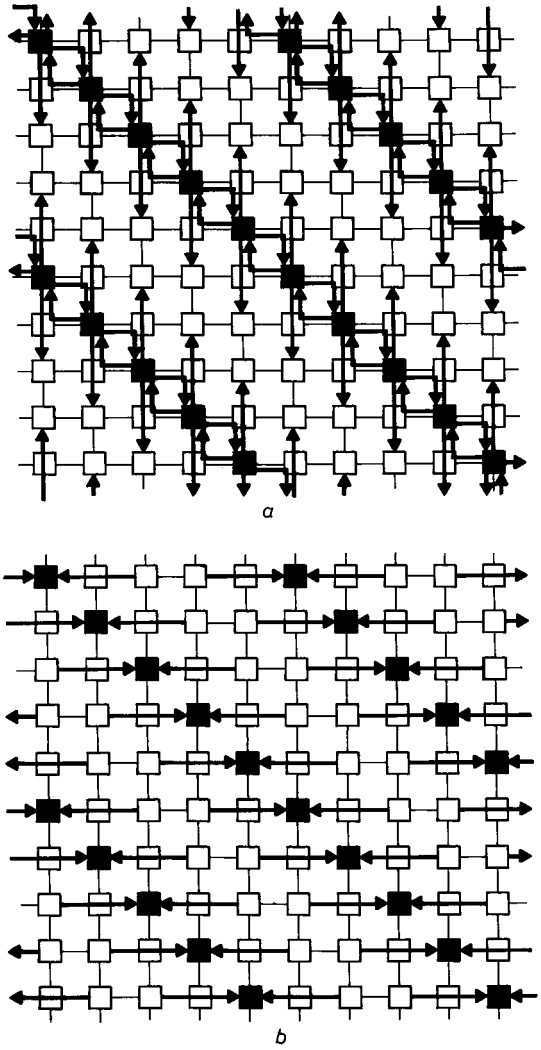

Fig. 11 *Data-movement operations*
a First data-movement operation
b Second data-movement operation

$\{9, 10, 11\}$ then $F = \{4, 0, 1\}$. Since this node is located in DDN_1, so the three messages should be moving from DDN_1 to DDN_0, DDN_1, and DDN_4. This task can be further accomplished as follows. Let $P_{k,k'}$ located in diagonal of DDN_i change destination list F as F', and F' is obtained as follows. For every $t \in F$, let $j = t - i$, if $j > h/2$ then let $j = j - h$ and $F' = F' \cup j$. For instance, for a destination list $F = \{4, 0, 1\}$, so $F' = \{-2, -1, 0\}$, where $i = 1$ and $h = 5$. Each element of F' represents the offset value from each DDN (See also Figs. 11 and 12). Further, the rest of the destination lists F' are displayed in Fig. 13.

(ii) **Data-movement operation**: Based on destination list F', a congestion-free data-movement operation is performed to balancing the load among all the DDNs. Suppose that $P_{k,k'}$ located in the diagonal in every DDN and each node has messages to be exchanged with nodes $P_{k+l,k'+l}$ when $l = \pm 1$ and ± 2. Every node $P_{k,k'}$ exchanges one message with node $P_{k+l,k'+l}$, $l = \pm 1$ and ± 2 within two time steps as shown in Fig. 10. Fig. 11 illustrate the first and second communication pattern. There is no

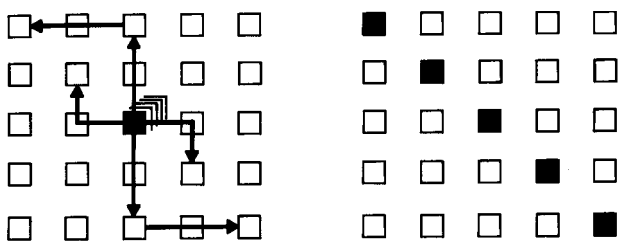

Fig. 10 *Data movement pattern*

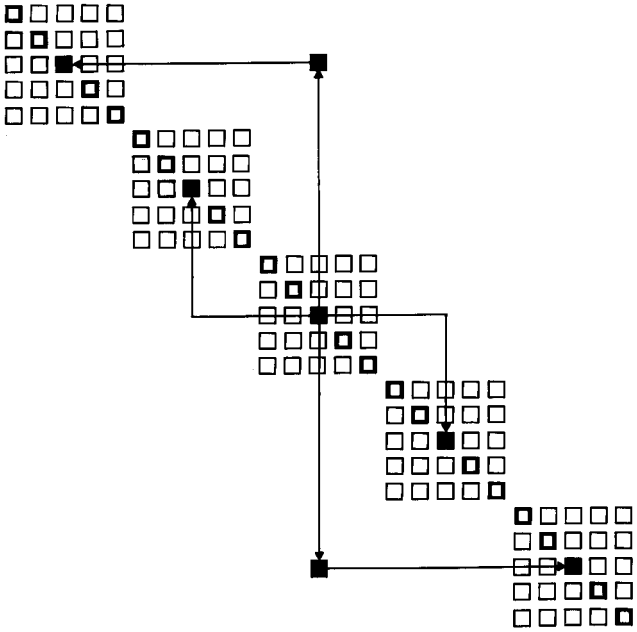

Fig. 12 *Data movement operation if $h > 5$*

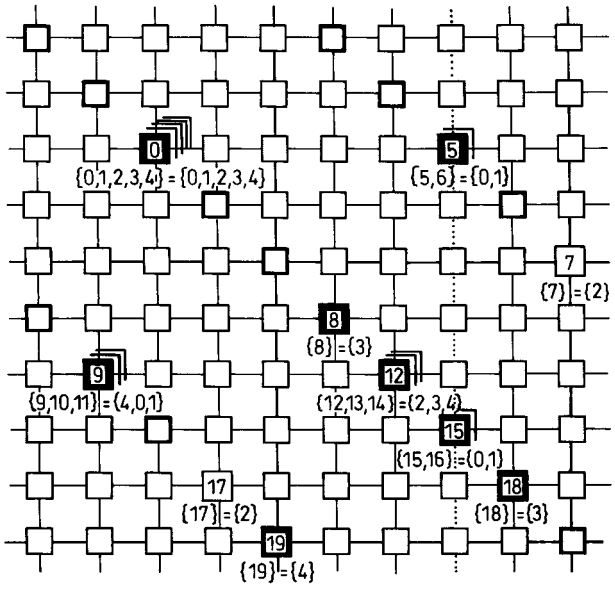

Fig. 13 *Final destination list*

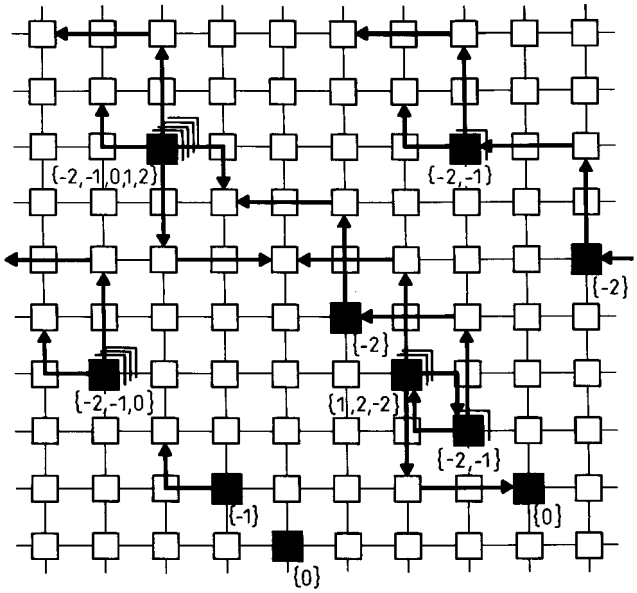

Fig. 14 *First data-movement operation*

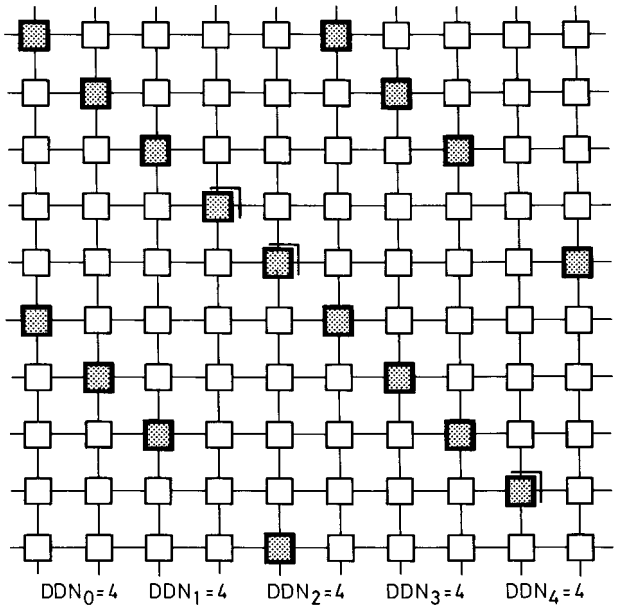

Fig. 15 *Second data-movement operation*

communication congestion. Each node $P_{k,k'}$ exchanges only one message with node $P_{k+l,k'+l}$, where $l \in F'$. Further, if $h > 5$, a hierarchial congestion-free communication pattern is also given in Fig. 12. Intuitively, it works within $2\lceil \log_5 h \rceil$ time steps. For instance, a data-movement operation is executed as illustrated in Figs. 14, 15 when $l = \pm 1$ and ± 2. Each communication stage only propagates message m at most; this stage needs time $2\lceil \log_5 h \rceil (T_s + mT_c)$. The time complexity of data tuning procedure is $2\lceil \log_5 h \rceil (T_s + mT_c)$.

3.2 Distribution phase

We now introduce the distribution phase. After executing the aggregation phase mentioned in Section 3.1, each *independent subnetwork* can have the same number of messages. These subnetworks, which are responsible for distributing the messages, can well exploit the communication parallelism and the characteristic of wormhole routing. The distribution phase is divided into three steps.

- *Alignment operation*: For each *DDN*, messages of source nodes located in the same *DDN* are collected into its main diagonal. A simple all-to-all broadcasting operation is then performed on the main diagonal to achieve the purpose of each node, which in the main diagonal contain the same number of messages.
- *Broadcast operation*: Each *DDN* performs dilated-diagonal broadcasting scheme [15].
- *Data-collection operation*: Each *DCN* collects all messages.

3.2.1 Alignment operation: After the aggregation phase, each *DDN* can keep the same number of messages indicated that each *DDN* will have equal communication latency. The purpose of the alignment phase lies in a preparation process for the distribution phase of multinode broadcasting. Two procedures are needed.

(i) **Alignment procedure to main diagonal**: All possible messages are collected into the main diagonal of the corresponding *DDN*. This task can be easily achieved by recursively performing the diagonal-based data redistribution operation as introduced in Section 3.1. Intuitively, it takes time $\lceil \log_5(\lceil n/h \rceil) \rceil (T_s + \tilde{m}T_c)$, where $\tilde{m} = sm/h$.

(ii) **All-to-all broadcasting procedure on diagonal**: This procedure is to collect messages of each node in the main diagonal from other nodes in the same diagonal. For instance, consider a $T_{5\times5}$, each node in the main diagonal of a *DDN* will keep a different message as shown in Fig. 16*a*. After executing the all-to-all-diagonal-broadcasting operation, all nodes in the main diagonal will keep messages from all other nodes as illustrated in Fig. 16*b*. The all-to-all-diagonal-broadcasting procedure is to recursively perform data-distribution and data-collection stages. We explain it by an example on a $T_{5\times5}$. The tasks is divided into two stages. In *data-distribution* each node $P_{i,i}$ in the diagonal distributes its own message to two neighbouring nodes $P_{i,i-2}$ and $P_{i,i+2}$ according to a data-distribution pattern as shown in Fig. 17*a*. In *data-collection* each node $P_{i,i}$ in diagonal then collect four messages from four neighbouring nodes $P_{i-1,i-1}$ and $P_{i+1,i+1}P_{i-2,i}$ and $P_{i+2,i}$ according to a data-collection pattern as illustrated in Fig. 17*b*. Therefore five nodes can contain all other nodes' messages in the same diagonal as shown in Figs. 17*c* and 16*b*. Repeating these data-redistribution and data-collection operations, an all-to-all-diagonal-broadcasting can be

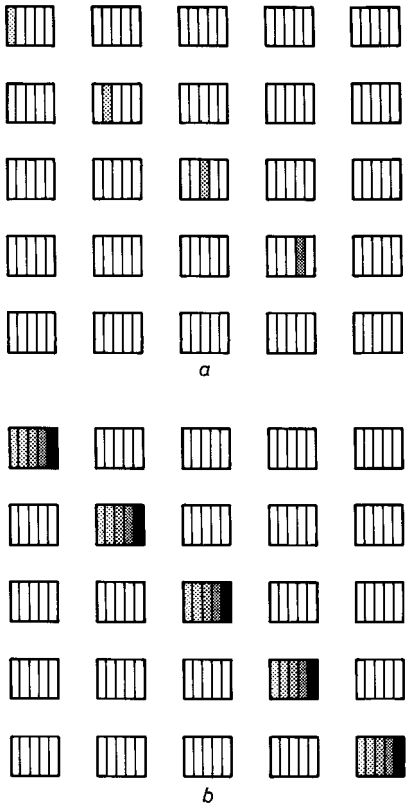

Fig. 16 *Alignment procedures*
a Nodes in main diagonal keep different messages
b All-to-all diagonal broadcast procedure

applied on the diagonal of *DDN* with any size. A larger example of a $T_{25\times 25}$ is illustrated in Fig. 18. Since data-distribution and data-collection stages needs time $2(T_s+\tilde{m}T_c)$, the time complexity is $2\lceil\log_5\lceil n/h\rceil\rceil(T_s+\tilde{m}T_c)$, where $\tilde{m}=sm/h$.

3.2.2 Broadcast operation: After the alignment operation, every node in the diagonal of each *DDN* contains the same broadcast message. The next step performs the dilated-diagonal broadcasting algorithm [15] on each *DDN* in parallel. In other words, we perform a dilated-diagonal broadcasting operation on each *DDN*. The main diagonal $D(P_{0,0},\lceil n/h\rceil)$ has all source packet's data, and the broadcasting is based on a recursive structure. The main diagonal sends messages to four diagonals and lets them also have the same messages. That is, each node of the main diagonal sends messages to four neighbouring diagonal nodes. The time complexity of broadcast phase is $\lceil\log_5\lceil n/h\rceil\rceil(T_s+\tilde{m}T_c)$, where $\tilde{m}=sm/h$.

3.2.3 Data collection operation: For each data collecting network (which is an $h\times h$ mesh) the diagonal nodes have received packets $(M_0, M_1,\ldots,M_{h-1})$. Every packet contains whole *DCN* messages. Each packet $(M_0, M_1,\ldots,M_{h-1})$ has messages. These messages should be propagated to every node of the *DCN*. This is implemented in two stages: row broadcasting followed by column broadcasting. The row broadcasting stage is done by applying a recursive scheme. We evenly partition *DCN* into three parts. The node located in a diagonal sends its own messages to two nodes located in the trisection part of the row of the *DCN* and recursively propagates to subtrisection until the distance is one. The row broadcasting pattern is shown in Fig. 19*a*. This takes $\lceil\log_3 h\rceil$ communication phases and incurs cost $T_1=\lceil\log_3 h\rceil(T_s+\tilde{m}T_c)$.

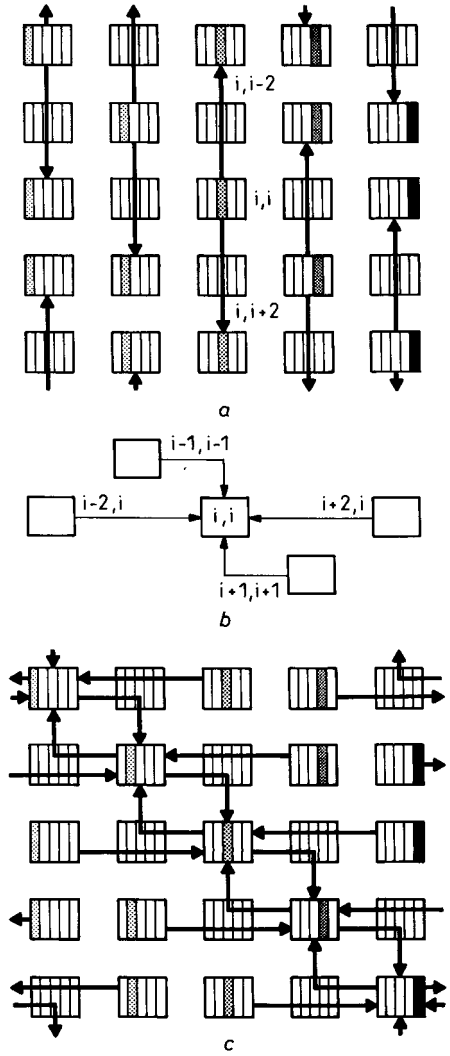

Fig. 17 *Alignment procedures*
a Data distribution pattern
b Data collection pattern
c Data collection stage

Every node collects the partial messages from the row broadcasting stage. The messages belong to its column nodes; every node will concurrently send separate message to other nodes with the pipelined scheme. We first embed a logical (directed) ring on each column of the *DCN*. This is done by first visiting even nodes down the column and then odd nodes up the column. Fig. 19*b* gives an example of such embedding. This gives a dilation-2 embedding. With this embedding, every node then pipeline propagates its own message following the ring of the $h\times h$ *DCN*. Figs. 20 and 21 illustrate the row broadcasting in a 10×10 2-D torus if $h=5$. The broadcasting result is shown in Fig. 22. The column broadcasting stage runs within $T_2=(h-1)(T_s+\tilde{m}T_c)$ Summing T_1 and T_2, the time complexity of data collecting is $T=T_1+T_2=(\lceil\log_3 h\rceil+h-1)(T_s+\tilde{m}T_c)$, where $\tilde{m}=sm/h$.

4 Performance analysis

We discuss the performance analysis of multinode broadcasting under two strategies; our proposed aggregation-then-distribution strategy and the well known result of edge-disjoint spanning trees (EDSTs) approach [5–7]. The time complexity of computation and communication latency of aggregation and distribution phases of our multinode broadcasting is given.

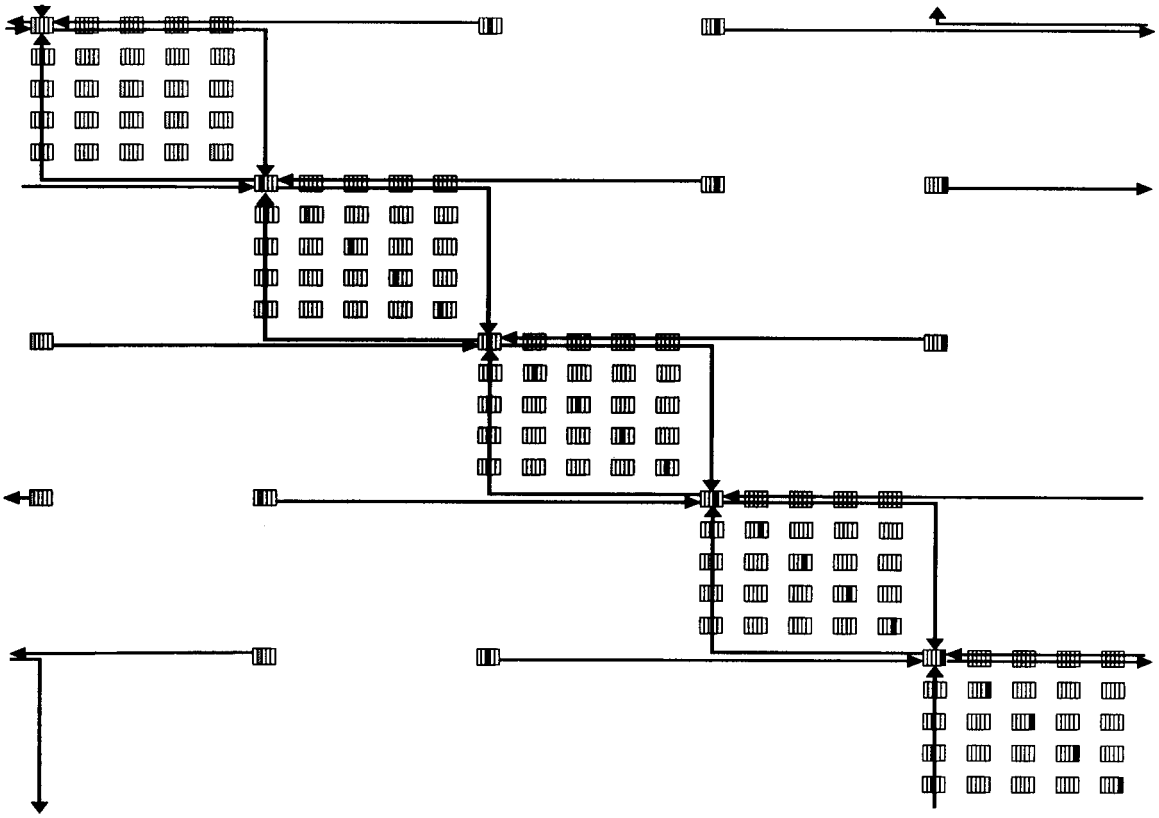

Fig. 18 *Larger example of all-to-all diagonal broadcast in T_{25*25}*

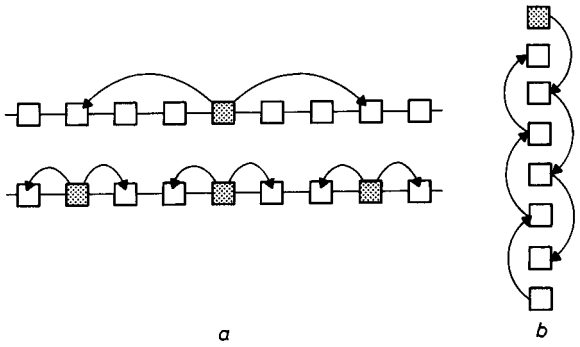

Fig. 19 *Data collection operation: row broadcasting*
a Row broadcasting pattern
b Dilated-2 ring embedded in column

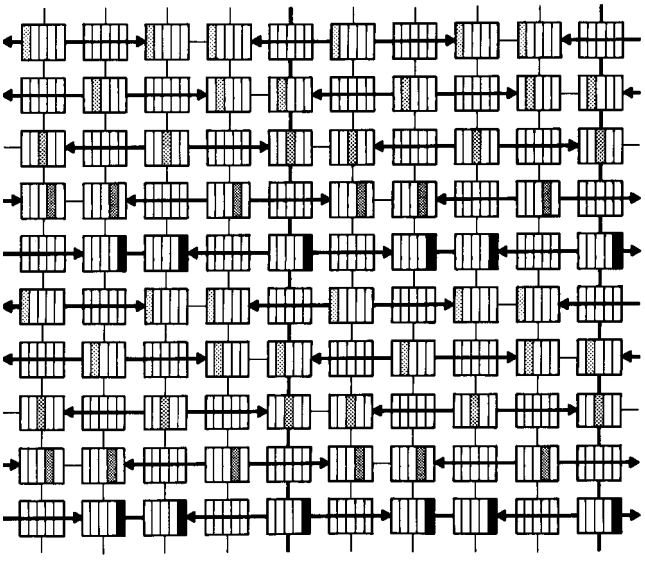

Fig. 20 *Row broadcasting in T_{10*10} 2D torus*

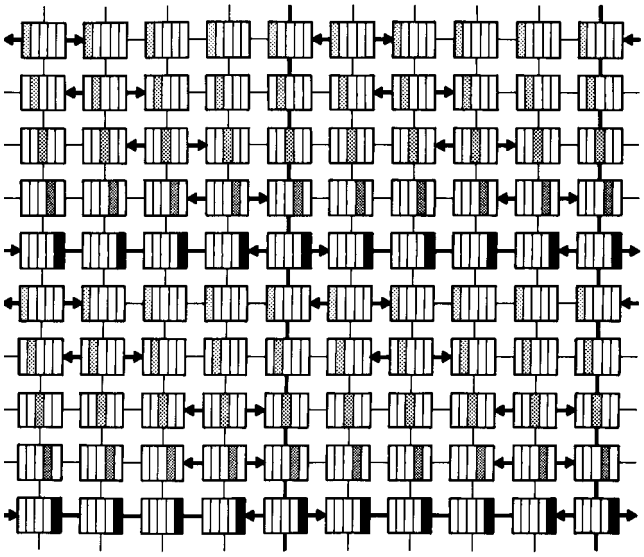

Fig. 21 *Recursive row broadcasting*

Lemma 2: The aggregation phase can be executed in a $T_{n\times n}$ torus within cost

$$O\left(\left(4\lceil\log_5 n\rceil + 3\lceil\log_5 h\rceil + 2\left\lceil\log_5\frac{n}{h}\right\rceil\right)T_s + \left[\left(2\lceil\log_5 h\rceil + \frac{5^{\lceil\log_5 h\rceil}-1}{4}\right)m + \left(4\lceil\log_5 n\rceil + 2\left\lceil\log_5\frac{n}{h}\right\rceil\right)\right]T_c\right)$$

where m is one unit message size.

Proof: The diagonal-based data-aggregation operation takes time $\lceil\log_5 h\rceil T_s + (5^{\lceil\log_5 h\rceil} - 1)/4mT_c$. The balan-

Table 1: Comparison of communication latency of multinode broadcasting using various schemes

Strategy	Start-up comp.	Trans. comp.
EDSTs [7]	$O(2\lfloor n/2\rfloor T_s)$	$O(2\lfloor n/2\rfloor \cdot sm/4T_c)$
Ours	$O(\max\{\lceil \log_5 n\rceil, h\}T_s)$	$O(\max\{\lceil \log_5 \lceil n/h\rceil\rceil, h\}s/hmT_c)$

cing-load operation takes time $(4\lceil \log_5 n\rceil + 2\lceil \log_5 n/h\rceil)(T_s + T_c) + 2\lceil \log_5 h\rceil(T_s + mT_c)$. Therefore the time complexity of aggregation phase is

$$O\left(\left(4\lceil \log_5 n\rceil + 3\lceil \log_5 h\rceil + 2\left\lceil \log_5 \frac{n}{h}\right\rceil\right)T_s + \left[\left(2\lceil \log_5 h\rceil + \frac{5^{\lceil \log_5 h\rceil}-1}{4}\right)m + \left(4\lceil \log_5 n\rceil + 2\left\lceil \log_5 \frac{n}{h}\right\rceil\right)\right]T_c\right).$$

Lemma 3: The distribution phase can be executed in a $T_{n\times n}$ torus within time

$$O\left(\left(4\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + \left\lceil \log_3 h\right\rceil + h - 1\right)T_s + \left(4\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + \left\lceil \log_3 h\right\rceil + h - 1\right)\tilde{m}T_c\right)$$

where $\tilde{m} = sm/h$.

Proof: The alignment operation needs time $3\lceil \log_5\lceil n/h\rceil\rceil(T_s + \tilde{m}T_c)$, where $\tilde{m} = sm/h$. The broadcasting operation takes time $\lceil \log_5\lceil n/h\rceil\rceil(T_s + \tilde{m}T_c)$. The data collection operation uses time $(\lceil \log_3 h\rceil + h - 1)(T_s + \tilde{m}T_c)$. Therefore the time complexity of distribution phase is

$$\left(4\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + \lceil \log_3 h\rceil + h - 1\right)T_s + \left(4\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + \lceil \log_3 h\rceil + h - 1\right)\tilde{m}T_c$$

where $\tilde{m} = sm/h$. □

Based on lemmas 2 and 3, the time complexity of the multinode broadcasting in 2-D torus using aggregation-then-distribution strategy is given.

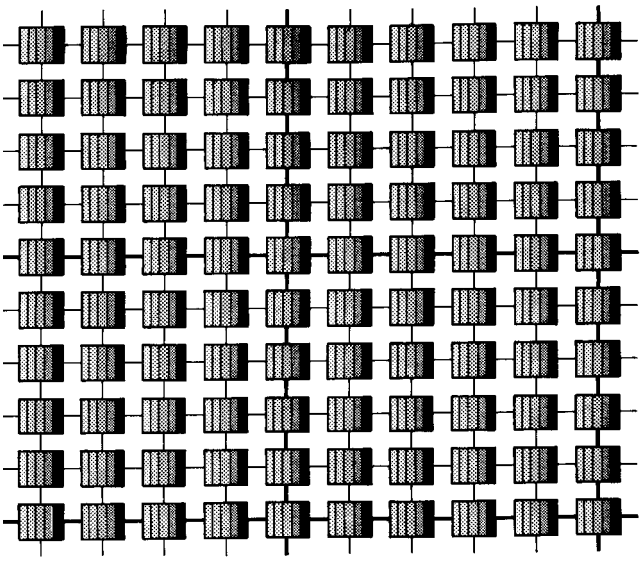

Fig. 22 *Broadcasting result*

Theorem 1: The multinode broadcasting algorithm with aggregation-then-distribution strategy can be executed in a $T_{n\times n}$ tours within

$$\begin{aligned} &O\left(\left(6\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + 3\lceil \log_5 h\rceil + \lceil \log_3 h\rceil + 4\lceil \log_5 n\rceil + h - 1\right)T_s\right. \\ &\quad + \left(4\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil + \lceil \log_3 h\rceil + h - 1\right)\tilde{m} \\ &\quad + \left(2\lceil \log_5 h\rceil + \frac{5^{\lceil \log_5 h\rceil}-1}{4}\right)m \\ &\quad \left. + \left(4\lceil \log_5 h\rceil + 2\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil\right)T_c\right) \\ &= O\left(\lceil \log_5 n\rceil T_s + \max\left(\left\lceil \log_5\left\lceil\frac{n}{h}\right\rceil\right\rceil, h\right)\tilde{m}T_c\right) \end{aligned}$$

where $\tilde{m} = sm/h$ and m is one unit message size.

Note that four edge-disjoint spanning trees can be constructed in a 2-D tori $T_{n\times n}$ [2], where the height of the spanning tree is $D+2$ and $D = 2\times\lfloor n/2\rfloor$ is the diameter in $T_{n\times n}$. The time complexity of multinode broadcasting in 2-D tori using edge-disjoint spanning trees approach is $O(\lfloor n/2\rfloor T_s + (2\lfloor n/2\rfloor \cdot sm/4)T_c)$ illustrated in Table 1. Table 1 reflects the fact that multinode broadcasting using aggregation-then-distribution strategy is more efficient than multinode broadcasting using the edge-disjoint spanning trees approach. The overall communication latency depends on the transmission complexity. Observe that the transmission complexity of our scheme is $O(\lceil \log_5\lceil n/h\rceil\rceil smT_c/h$. The transmission complexity of our scheme is better than the edge-disjoint spanning trees-based approach owing to $O(\lceil \log_5\lceil n/h\rceil\rceil smT_c/h) < O(2\lfloor n/2\rfloor \cdot sm/4)T_c)$. We conclude this Section with the observation that the multinode broadcasting using our aggregation-then-distribution strategy is truly more efficient than multinode broadcasting using edge-disjoint spanning trees approach.

5 Conclusions

We have shown how to solve the multinode broadcast problem in a 2-D torus using an aggregation-then-distribution strategy. The underlying assumptions are wormhole and dimension-ordered routing, as currently used. The main technique is to partition the torus into independent subnetworks. Timing analyses have shown that this scheme is promising. Work is currently underway to develop multinode broadcasting of personalised messages and to extend to higher dimensional tori and other networks.

6 Acknowledgments

This work was supported by the National Science Council, R.O.C., under contract NSC89-2218-E-305-002.

7 References

1 ROBINSON, D.F., MCKINLEY, P.K., and CHENG, B.H.: 'Optimal multicast communication in wormhole-routed torus networks', *IEEE Trans. Parallel Distrib. Syst.*, 1995, **6**, (10), pp. 1029–1042

2 KESAVAN, R., and PANDA, D.K.: 'Multiple multicast with minimized node contention on wormhole *k*-ary *n*-cube networks', *IEEE Trans. Parallel Distrib. Syst.*, 1999, **10**, (4), pp. 371–393

3 SAAD, Y., and SCHULTZ, M.: 'Data Communication in hypercubes', *J. Parallel Distrib. Comput.*, 1989, **6**, (1), pp. 115–135

4 SAAD, Y., and SCHULTZ, M.: 'Data communication in parallel architectures', *Parallel Comput.*, 1989, **11**, (5), pp. 131–150

5 STAMOULIS, G.D., and TSITSIKLIS, J.N.: 'An efficient algorithm for multiple simultaneous broadcasts in the hypercube', *Inf. Process. Lett.*, 1989, **46**, (12), pp. 219–224

6 TSENG, Y.C.: 'Multi-node broadcasting in hypercubes and star graph', *J. Inf. Sci. Eng.*, 1998, **14**, (4), pp. 809–820

7 VARVARIGOS, E.A., and BERTSEKAS, D.P.: 'Partial multinode broadcast and partial exchange algorithms for d-dimension meshes', *J. Parallel Distrib. Comput.*, 1994, **23**, (5), pp. 177–189

8 HAMBRUSCH, S.E., KHOKHAR, A.A., and LIN, Y.: 'Scalable S-to-P broadcasting on message-passing MPPs', *IEEE Trans. Parallel Distrib. Syst.*, 1998, **9**, (8), pp. 758–768

9 TSENG, Y.C., CHEN, Y.S., JUANG, T.Y., and CHANG, C.J.: 'Congestion-free, dilation-2 embeding of complete binary tree in star graphs', *Networks*, 1999, **33**, (3), pp. 221–231

10 COSNARD, M., and TRYSTRAM, D.: 'Parallel algorithms and architectures' (Thomson Computer Press, Boston, MA, 1995)

11 CHEN, Y.S., JUANG, T.Y., and TSENG, E.H.: 'Efficient broadcast in an arrangement graph using multiple spanning trees', *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, 2000, **E83-A**, (1), pp. 139–149

12 TSENG, Y.C., WANG, S.Y., and HO, C.W.: 'Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach', *IEEE Trans. Parallel Distrib. Syst.*, 1999, **10**, (1), pp. 44–61

13 LEIGHTON, F.T.: 'Introduction to parallel algorithms and architectures: arrays-trees-hypercube' (Morgan Kaufmann, San Mateo, CA, 1992)

14 BRUCK, J., CYPHER, R., and HO, C.T.: 'Fault-tolerant de Bruijn and shuffle-exchange networks', *IEEE Trans. Parallel Distrib. Syst.*, 1994, **5**, (5), pp. 548–553

15 TSENG, Y.C.: 'A dilated-diagonal-based scheme for broadcast in a wormhole-routed 2D torus', *IEEE Trans.*, 1997, **C-46**, (8), pp. 947–952