Path-Based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing

Masoumeh Ebrahimi, *Member*, *IEEE*, Masoud Daneshtalab, *Member*, *IEEE*, Pasi Liljeberg, *Member*, *IEEE*, Juha Plosila, *Member*, *IEEE*, José Flich, *Member*, *IEEE*, and Hannu Tenhunen, *Member*, *IEEE*

Abstract—Combining the benefits of 3D ICs and Networks-on-Chip (NoCs) schemes provides a significant performance gain in Chip Multiprocessors (CMPs) architectures. As multicast communication is commonly used in cache coherence protocols for CMPs and in various parallel applications, the performance of these systems can be significantly improved if multicast operations are supported at the hardware level. In this paper, we present several partitioning methods for the path-based multicast approach in 3D mesh-based NoCs, each with different levels of efficiency. In addition, we develop novel analytical models for unicast and multicast traffic to explore the efficiency of each approach. In order to distribute the unicast and multicast traffic more efficiently over the network, we propose the Minimal and Adaptive Routing (MAR) algorithm for the presented partitioning methods. The analytical and experimental results show that an advantageous method named Recursive Partitioning (RP) outperforms the other approaches. RP recursively partitions the network until all partitions contain a comparable number of switches and thus the multicast traffic is equally distributed among several subsets and the network latency is considerably decreased. The simulation results reveal that the RP method can achieve performance improvement across all workloads while performance can be further improved by utilizing the MAR algorithm. Nineteen percent average and 42 percent maximum latency reduction are obtained on SPLASH-2 and PARSEC benchmarks running on a 64-core CMP.

Index Terms—3D Networks-on-Chip, unicast and multicast communication, partitioning methods, analytical models, adaptive routing algorithm

1 Introduction

Networks-on-chip (NoCs) have been proposed as a promising solution for designing the interconnect fabric of multicore systems [1], [2], [3]. Planar (2D) chip fabrication technology is facing new challenges in the deep submicron regime [4], [5]. Wire delay and power consumption increase significantly by the usage of global interconnects in 2D designs. To overcome these limitations, technology is moving rapidly toward the concept of 3D ICs where multiple active silicon layers are vertically stacked. The major advantages of 3D NoCs are the considerable reduction in the average wire length and wire delay, resulting in lower power consumption and higher performance [5], [6], [7], [8], [9].

Unicast and multicast communication can be considered for a NoC. In the unicast communication case, a message is sent from a source node to a single destination node, while in the multicast communication, a message is delivered from one source node to an arbitrary number of destinations. Multicast can be easily implemented with no hardware overhead by assuming a multicast message is replicated and every instance is sent to a particular destination (this is termed unicast-based multicast). However, this implementation is inefficient. This inefficiency arises because sending multiple copies of the same message into the network not only causes a significant amount of traffic, but also introduces a large serialization delay at the injection point. The vast majority of traffic in Multiprocessor Systems-on-Chip (MPSoCs) consists of unicast traffic and most studies have assumed that the traffic is only unicast. Thereby, the concept of unicast communication has been studied extensively in the literature. The proposed unicast protocols are efficient when all injected messages are unicast. However, if only a small percentage of the total traffic is multicast, the efficiency of the overall system is considerably reduced. Indeed, multicast communication has a large impact on Chip Multiprocessor (CMP) systems performance. The multicast communication is frequently present in many cache coherence protocols (e.g., directory-based protocols, token-based protocols, and Intel QPI protocol [10], [11]). For example, around 5 percent of total traffic in a SGI-Origin protocol (which is a directory-based protocol) consists of multicast messages [11]. In this protocol, message latency can be reduced by 50 percent, if multicast is supported in hardware, thus

- *M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen are with the Department of Information Technology, University of Turku, Joukahaisenkatu 3-5B, 20520 Turku, Finland. E-mail: {Masoumeh.ebrahimi, masoud.daneshtalab, pasi.liljeberg, juha.plosila, Hannu.tenhunen}@utu.fi.*
- *J. Flich is with the Departamento de Informática de Sistemas y Computadores (DISC), Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Valencia, 2 piso, despacho 2N-4, Apdo 22012, 46071 Valencia, Spain. E-mail: jflich@disca.upv.es.*

Copyright text

highlighting the importance of hardware-level multicast support. In this paper, we performed some analysis to determine the percentage of multicast messages generated by coherence protocols. We analyzed synthetic and application traces (i.e., SPLASH-2 [12], PARSEC [13], [14]) on top of two popular coherence protocols, MESI [15] and token-based MOESI [16] (the detailed system configuration parameters and workloads can be found in Table 3). Based on experimental results, the bulk of the traffic in MESI protocol is generated by unicast messages while token-based MOESI protocol is heavily multicast based. On account of our analysis, on average, around 10 percent of MESI traffic and more than 80 percent of token-based MOESI traffic are multicast.

Hardware-based multicast schemes can be broadly classified into path-based [18], [19], [20] and tree-based [18], [19] methods. In the tree-based method, a spanning tree is built at the source and a multicast message is sent down the tree. The source is considered as the root while destinations are the leaves of this tree. The message is replicated along its route at switches and forwarded along multiple outgoing channels reaching to disjoint subsets of destinations [3]. In the path-based multicast method, the switch prepares a message for delivery to a set of destinations by placing the list of destinations in the header of the message. The message is routed along the path until it reaches the first destination. The message is delivered both to the local core and to the corresponding output channel for continuing the path toward the next destination in the list. In this way, the message is eventually delivered to all specified destinations. A number of studies have shown that path-based methods exhibit superior performance characteristics over tree-based counterparts [21], [22]. The path-based approach does not replicate messages within the network, thus not increasing message contention. However, the path visiting all switches can become large. To reduce the length of the multicast path, destinations can be divided into several disjoint subsets at the network interface of the source switch, and then copies of the message are sent across several separate multicast paths with different destination sets [3]. Partitioning methods try to reduce latency and increase the performance via an efficient partitioning of destinations into disjoint subsets [33].

Additionally, routing algorithms can be classified into deterministic or adaptive algorithms. A deterministic routing algorithm uses a fixed path for each pair of switches resulting in increased network latency especially in congested networks. In contrast, in adaptive routing, a message is not restricted to a single path while traveling from a source switch to its destination(s). A message may take a different output at a given switch because the other paths are congested. Therefore, adaptive routing algorithms can obtain better performance [23], [24], [25].

In this paper, we tackle how to efficiently implement routing in 3D mesh-based NoCs, addressing both unicast and multicast traffic. To do this, we present several partitioning methods, named Two-Block Partitioning (TBP), Vertical-Block Partitioning (VBP), and Recursive partitioning method (RP), for the path-based multicast approach, each with different level of efficiency. In Two-Block Partitioning method, destinations are divided into two groups and a multicast message is responsible to deliver the message to all destinations within each group. This algorithm performs well when the network size is small. However, as the network enlarges, a message may take a long path to deliver the multicast message to all destinations and thus increasing latency. In Vertical-Block Partitioning method, destinations are divided into more number of groups depending on their vertical columns. This method suggests a better degree of parallelism and lower latency as a message is dedicated to a smaller set of destination switches and thus a shorter path is taken by each message. The main disadvantage of this method is regarding to the creation of unbalanced partitions as a group may contain a large set of switches compared with others. This results in taking long paths by some messages and short paths by others, keeping the multicast latency still high. Recursive partitioning method tries to have the comparable number of switches within each partition while keeping the number of messages low. This results in lower average latency compared with TBP and VBP methods. To explore the efficiency of each approach, in addition to simulation experiments, we develop novel analytical models for unicast and multicast traffic. The analytical and experimental results show that RP outperforms the other approaches. On top of all partitioning methods, and in order to efficiently distribute the unicast and multicast messages, we design a minimal and adaptive routing algorithm, named MAR, based on the Hamiltonian path for all partitioning methods. The algorithm is simple and does not require any virtual channel for neither unicast nor multicast messages. The main properties of the final approach which is a combination of the RP and MAR methods can be summarized as follows: 1) decreasing the latency of messages by addressing the nonoptimal solutions of ordinary partitioning methods; 2) alleviating the traffic congestion by presenting an adaptive routing algorithm for both unicast and multicast messages; and 3) causing a relatively small area overhead mainly by not using virtual channels for deadlock avoidance and providing a simple implementation of the routing algorithm.

The rest of this paper is organized as follows: Section 2 reviews related work. A brief background on the Hamiltonian path strategy along with the proposed partitioning methods is discussed in Section 3. The minimal adaptive routing is presented in Section 4. The results are given in Section 5 while we summarize and conclude in the last section.

2 Related Work

Due to the fact that the multicast communication is used commonly in various parallel applications, there have been several attempts to improve the performance of multicast communication in 2D NoCs. "Virtual Circuit Tree Multicasting" (VCTM) [10], "Recursive Partitioning Multicast" (RPM) [27], and "Hamiltonian path-based multicast algorithm for NoCs" [19], [20] are three recent works focused on 2D NoCs. VCTM and RPM are tree-based methods and the proposed algorithms in [19], [20] are path-based methods. In VCTM method, when the number of destinations is high, a

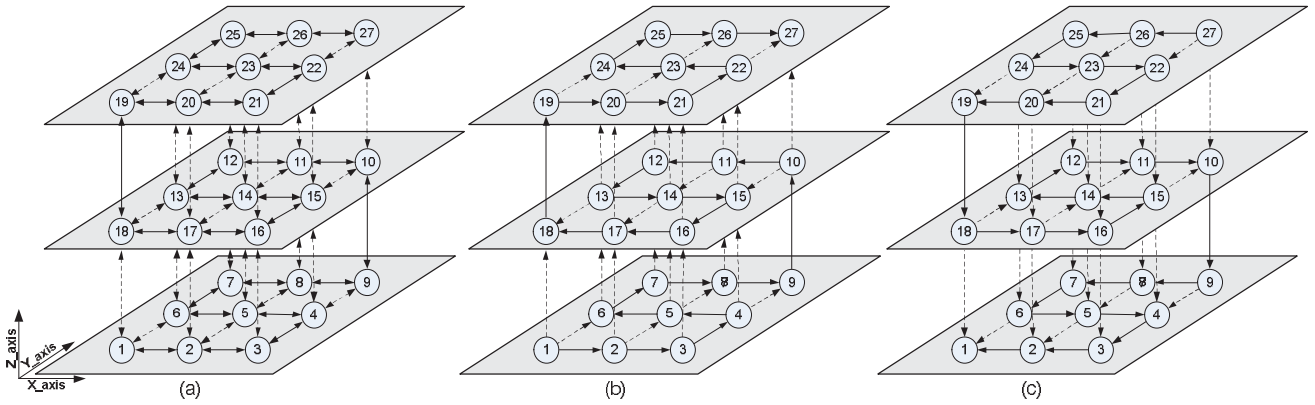


Fig. 1. (a) A $3 \times 3 \times 3$ mesh physical network with the label assignment, (b) high channel, (c) low channel subnetworks. The solid lines indicate the Hamiltonian path and dashed lines indicate the links that could be used to reduce path length in routing.

large number of setup messages must be delivered into the network (before delivering the real multicast message) which decreases performance significantly. The area overhead of VCTM is relatively high due to maintaining a table at each switch to store the information of a virtual circuit tree. In RPM method, the processing of the header information is complex and performed several times for each multicast message. The common disadvantage of VCTM and RPM method is that a message may hold several channels for extended periods of time to receive all requested output channels, thereby increasing network contention [3]. Finally, both RPM and VCTM methods are based on deterministic algorithms and cannot provide adaptiveness to neither unicast nor multicast messages.

A solution to overcome the disadvantages of tree-based multicast is to utilize path-based multicast routing. The authors in [19] present a deadlock-free adaptation of the dual-path multicast algorithm for 2D mesh NoCs and evaluate the performance impact, demonstrating the efficiency of the proposed multicast algorithm. However, this method cannot provide any adaptiveness for routing unicast/multicast messages and has a disadvantage of high network latency due the creation of long paths in the network which has been improved in [20]. To the best of our knowledge, there has not been any prior study on path-based multicast routing in 3D NoCs. However, some related studies can be found in the multicomputer domain [28], [29], [30]. An adaptive multicast communication in 3D mesh networks is discussed in [28]. The algorithm is based on an extension of a theory defined in [29] from 2D to 3D mesh networks. The algorithm utilizes the Hamiltonian path but provides adaptiveness and prevents deadlocks by using virtual channels. However, adding virtual channels is costly in NoCs due to increased arbitration complexity and buffering requirements [31]. Two additional methods of unicast/multicast communication in 3D mesh-based networks are presented in [29] and [30]. The proposed methods are guaranteed to be deadlock-free by means of the Hamiltonian path. However, the presented algorithms are deterministic and suffer from low performance and their inability to partition the network efficiently.

3 Partitioning Methods

The performance of a multicast operation can be measured in terms of its latency in delivering a message to all its destinations. Multicast latency consists of two components: the startup latency and the network latency. The startup latency (startup-latency; SL) is the time required to create several messages (each with a different set of destinations), prepare the messages, and start injecting them into the network. The network latency for multicast messages is defined as the time elapsed from the first flit injection into the network to the reception of the last flit in all destinations of the multicast message. Based on that, we define the mean network multicast latency (mean-mul-latency; MML) and the maximum network multicast latency (max-mul-latency; MxML).

As previously commented, partitioning methods help in reducing the network latency component [33]. In particular, these methods divide the network into several logical partitions and assign destinations to different sets, one set for each partition and including destinations that belong to that partition. Smart partitioning methods must balance the sets and reduce the path length within each partition. However, breaking the network into logical partitions may have the following deficiencies: 1) a large number of network partitions will lead to additional latency as more startup messages (SM) will need to be prepared at the source node and this latency is usually high. 2) an unbalanced configuration of partitions will create long paths within the network. In both cases, the latency of the multicast operation will be increased.

3.1 Hamiltonian Path

The Hamiltonian path strategy [18] guarantees that the network will be free of deadlocks for both unicast and multicast traffic. As shown in Fig. 1a, for each switch, a label is assigned from 1 to N where N is the number of switches in the network. A Hamiltonian path visits all the switches[1] and each switch is visited exactly once. Several Hamiltonian paths can be considered in the mesh topology. In $a \times b \times c$ mesh network, each switch is labeled by an integer value according to its x, y, and z coordinates. The following

1. For the sake of understanding, we assume the node X is connected to switch X and the labels are for switches, not nodes.

equations show one possibility of assigning the labels, which we utilize in this paper:

$$L(x,y,z) = \{(a \times b \times z) + (a \times y) + (x+1)\} \quad where\ z: even,\ y: enen,$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times y) + (a-x)\} \quad where\ z: even,\ y: odd,$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times (b-y-1)) + (a-x)\} \quad where\ z: odd,\ y: even,$$
$$L(x,y,z) = \{(a \times b \times z) + (a \times (b-y-1)) + (x+1)\} \quad where\ z: odd,\ y: odd.$$

As exhibited in Fig. 1, two directed Hamiltonian paths (or two subnetworks) are constructed by this labeling. The high channel subnetwork starts at switch 1 (Fig. 1b), and the low channel subnetwork ends at switch 1 (Fig. 1c). In case the label of the destination switch is greater than the label of the source switch, the routing always takes place in the high channel subnetwork (Fig. 1b); otherwise it takes place in the low channel subnetwork (Fig. 1c). Notice that there are shortcut channels (those drawn in dashed lines) that do not take part in the Hamiltonian path. However, shortcut channels can be used by messages to improve performance. In this case, half of those channels are used for the high channel subnetwork and half for the other subnetwork. Deadlock is prevented as two separate sets of resources are used for each direction and messages never change their directions. Thus, no dependencies between the two sets are introduced.

3.2 Partitioning Methods Based on the Hamiltonian Path Strategy

In the partitioning methods, the destinations are grouped in two sets at each source. One set includes all the destinations that are reached using the high channel subnetwork, and the other set includes the remaining destinations reached using the low channel subnetwork.

In the next section, we explain the TBP method in detail, and then we introduce two other partitioning methods, VBP and RP. Notice that the TBP method is a straight forward extension of the dual-path multicast in 2D NoCs [19] to 3D NoCs. It can be seen as a naïve method since no effort is made to balance the two sets. For each partitioning method, we provide some analysis on the number of startup messages, latency of multicast operations (MML and MxML), and the average latency of unicast operations (AUL). Analytical models are provided for unicast and multicast messages assuming zero-load latency [9], [32]. Based on the zero-load latency, a message never contends for network resources with other messages. Under this assumption, the performance of each approach can be measured based on the number of hops required for delivering a message from a source node to its destination(s). Contention effects will be accounted analytically in addition to experiments with our simulation platform.

3.2.1 Two-Block Partitioning

The Two-Block Partitioning method is a base scheme in which all switches are split in two disjoint sets: a high set and a low set. As shown in Fig. 2, when considering the label assignment of the Hamiltonian path strategy, all switches located in the same 2D layer as the source switch

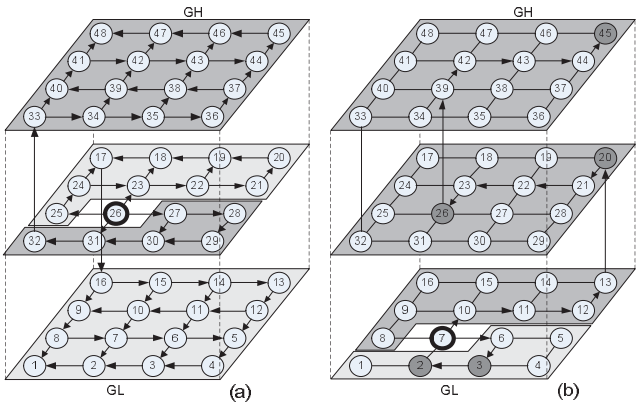


Fig. 2. The TBP method (a) balanced, (b) unbalanced partitions.

are distributed between the two sets while all the switches in higher or lower 2D layers are put in the high or low sets, respectively. In addition, when multicasting, at maximum one message is created for each set and the destinations within each set are reached according to the Hamiltonian label. Therefore, destinations in the high set are visited in ascending order and destinations in the low set are visited in descending order.

Fig. 2a shows an example of the TBP partitioning policy and the portions of each partition that depends on the source switch position. As illustrated in this figure, if the source switch is located in a middle layer, two partitions cover comparable numbers of switches but still with a large number of switches in both partitions. However in Fig. 2b, one partition contains considerably more switches than the other. Now, suppose that the multicast message $m = (7, \{2, 3, 20, 26, 45\})$ is generated at switch 7. Destination IDs are split into two sets and should be visited accordingly to their labels: $G_H = \{20, 26, 45\}$ and $G_L = \{3, 2\}$. The message created for GH uses the Hamiltonian path as follows: *{7,10,11,12,13,**20**,21,22,23,**26**,39,42,43,44,**45**}* where 14 hops are needed to reach the last destination. The message path for the GL is: *{7,6,**3**,**2**}* where three hops are required for delivering the message to all destinations in the low channel subnetwork. In the TBP method, the number of startup messages is low and never gets larger than two. However, it suffers from high network latency due to unbalanced partitions and high probability of long paths within the network.

Avg-Uni-Latency (AUL). Since messages can utilize shortcut channels without introducing new cycles, the path taken by unicast messages is reduced to minimal paths between each pair of source and destinations. Assuming uniform distribution of destinations and using minimal paths for unicast messages, the average unicast latency for $a \times b \times c$ 3D mesh-based network is [9]:

$$\mathrm{AUL_{3D}} = \frac{\mathrm{a^2bc + ab^2c + abc^2 - ac - bc - ab}}{\mathrm{3abc}}. \qquad (1)$$

Regardless of the partitioning method used, unicast messages are routed within the network in the same manner, so (1) is valid for the VBP and RP methods. This equation can be easily applied to 1D (when $b = 1$ and $c = 1$) and 2D (when $c = 1$) mesh networks.

The startup latency and network latency for multicast messages. The multicast latency depends on the number and the location of destinations. This makes computing the analytical multicast latency complex. In order to simplify the complexity, we consider that the latency of a multicast message is set by the final destination so that the multicast message always takes the longest path within the network (without using shortcut channels). This is the worst case. For instance, in the previous example, the two messages have their final destinations set as 45 and 2, and their distances from the source switch are 14 and 3 hops, respectively. However, in MML, we consider the longest path from the source to destinations 45 and 2 which are 41 and 6 hops, respectively. In the TBP method, the path between two destinations to reach in a sequential order is minimal while the path from the source to each destination is not necessarily minimal. As an example in Fig. 2, the paths from switch 7 to 20, 20 to 26, and 26 to 45 are minimal; however, the paths from switch 7 to 26 and 7 to 45 are nonminimal. According to this, MML for every switch j in $a \times b \times c$ network is (Where n is the total number of switches in the network.)

$$\mathrm{MML_j} = \frac{1}{\mathrm{n}} \left(\sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{j}-1} \mathrm{i} + \sum_{\mathrm{i}=\mathrm{j}+1}^{\mathrm{i}=\mathrm{n}} (\mathrm{i}-\mathrm{j}) \right). \tag{2}$$

The average multicast latency for the whole network in the TBP method can be obtained by

$$\begin{aligned} \mathrm{MML_{TBP}} &= \frac{1}{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} \mathrm{MML_j} \\ &= \frac{1}{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} \frac{1}{\mathrm{n}} \left(\sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{j}-1} \mathrm{i} + \sum_{\mathrm{i}=\mathrm{j}+1}^{\mathrm{i}=\mathrm{n}} (\mathrm{i}-\mathrm{j}) \right) \Bigg) = \frac{\mathrm{n}^2-1}{3\mathrm{n}}. \end{aligned} \tag{3}$$

This equation is proved by using the following set of formulas. The sum of partial factorial formula is given by

$$\begin{aligned} &\frac{\mathrm{m}!}{0!} + \frac{(\mathrm{m}+1)!}{1!} + \frac{(\mathrm{m}+2)!}{2!} + \cdots + \frac{(\mathrm{m}+\mathrm{n}-1)!}{(\mathrm{n}-1)!} \\ &\quad = \frac{(\mathrm{m}+\mathrm{n})!}{(\mathrm{m}+1)(\mathrm{n}-1)!}. \end{aligned} \tag{4}$$

For all positive integers, we get the formula when $m = 1$ or $m = 2$:

$$1 + 2 + 3 + \cdots + \mathrm{n} = \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{n}} \mathrm{i} = \frac{\mathrm{n}(\mathrm{n}+1)}{2} \tag{5}$$

$$\begin{aligned} &1 \times 2 + 2 \times 3 + 3 \times 4 + \cdots + \mathrm{n} \times (\mathrm{n}+1) \\ &\quad = \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{n}} \mathrm{i}(\mathrm{i}+1) = \frac{\mathrm{n}(\mathrm{n}+1)(\mathrm{n}+2)}{3}. \end{aligned} \tag{6}$$

By using (5) and (6), $\mathrm{MML_{TBP}}$ can be written as follow and (3) is proved:

$$\begin{aligned} \mathrm{MML_{TBP}} &= \frac{1}{2\mathrm{n}^2} \left(\sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} (\mathrm{j}-1)(\mathrm{j}) + \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} (\mathrm{n}-\mathrm{j})(\mathrm{n}-\mathrm{j}+1) \right) \\ &= \frac{(\mathrm{n}-1)(\mathrm{n}+1)}{3\mathrm{n}} = \frac{\mathrm{n}^2-1}{3\mathrm{n}}. \end{aligned} \tag{7}$$

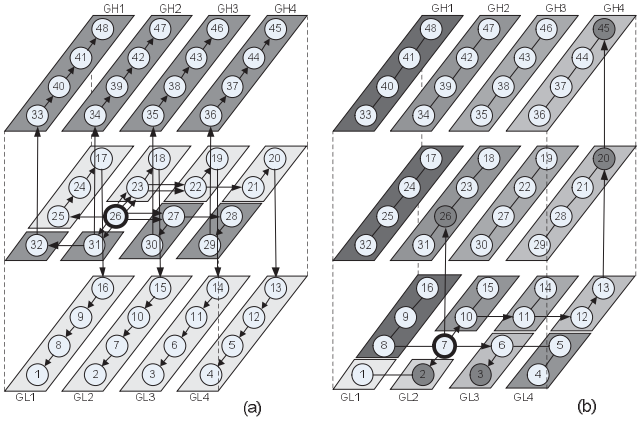


Fig. 3. The VBP method (a) balanced, (b) unbalanced partitions.

MxML is the time when a multicast operation is completed and a message reaches all its destinations. The MxML for a source switch j and the whole network are given by

$$\mathrm{MxML_j} = \begin{cases} \mathrm{n}-\mathrm{j} & \text{if } 0 \le \mathrm{j} \le \left\lfloor \frac{\mathrm{n}}{2} \right\rfloor \\ \mathrm{j}-1 & \text{if } \left\lfloor \frac{\mathrm{n}}{2} \right\rfloor \le \mathrm{j} \le \mathrm{n}, \end{cases} \tag{8}$$

$$\mathrm{MxML_{TBP}} = \frac{2}{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}/2} (\mathrm{n}-\mathrm{j}) = \begin{cases} \frac{3\mathrm{n}-2}{4} & \text{if n : even} \\ \frac{3\mathrm{n}^2-2\mathrm{n}-1}{4\mathrm{n}} & \text{if n : odd.} \end{cases} \tag{9}$$

In TBP, destinations are split in two sets. Thus, maximum startup messages is set to two regardless of the source switch location. There are two exceptions regarding the first and last switches which can deliver only one multicast message to the network.

3.2.2 Vertical Block Partitioning

In this method, similar to the TBP method, the network is partitioned into high and low channel subnetworks. Destinations are divided into high and low sets. In an additional step, each subnetwork is vertically partitioned such that switches in the same column (with the same a value in $a \times b \times c$ network) are included in a new set.

As illustrated in Fig. 3, this scheme does not guarantee balanced partitions. For the switch located at 26, partitions are balanced, but they are not balanced when the source is at switch 7 (i.e., four subnetworks cover more switches than the others). Moreover, the time required to prepare at most eight messages is considered as the number of startup messages. For the multicast message $m = (7, \{2, 3, 20, 26, 45\})$, four sets are formed: $G_{H2} = \{26\}$, $G_{H4} = \{20, 45\}$, $G_{L2} = \{2\}$, and $G_{L3} = \{3\}$. One message is generated for each set and message paths are *{7,**26**},{7,10,11,12,13,**20**,**45**},{7,**2**}* and *{7,6,**3**}* where the maximum hop count is six.

This scheme has several advantages over the TBP method as it achieves a high level of parallelism; avoids the creation of long paths and reduces the network latency. The VBP method increases, however, the number of startup messages as it requires up to 2a messages in $a \times b \times c$ network. In addition, this scheme does not guarantee

balanced partitions as it is balanced only when the source switch is located in a middle layer while some partitions may cover considerably more switches than the others when the source switch is located at the top or bottom layer.

The startup latency and network latency for multicast messages. Since the network is symmetric and is partitioned vertically, the MML value can be measured in one vertical partition and then generalized to other partitions. For this purpose, we consider that $a \times b \times c$ mesh network is divided into a vertical partitions where each partition contains bc switches. Using (2) and (3), the MML value for a source switch j inside a vertical partition and for all switches in a partition can be computed as follows:

$$\mathrm{MML_j} = \frac{1}{\mathrm{bc}} \left(\sum_{\mathrm{i=1}}^{\mathrm{i=j-1}} \mathrm{i} + \sum_{\mathrm{i=j+1}}^{\mathrm{i=bc}} (\mathrm{i-j}) \right), \tag{10}$$

$$\begin{aligned} \mathrm{MML_{bc}} &= \frac{1}{\mathrm{bc}} \sum_{\mathrm{j=1}}^{\mathrm{j=bc}} \frac{1}{\mathrm{bc}} \left(\sum_{\mathrm{i=1}}^{\mathrm{i=j-1}} \mathrm{i} + \sum_{\mathrm{i=j+1}}^{\mathrm{i=bc}} (\mathrm{i-j}) \right) \\ &= \frac{(\mathrm{bc})^2 - 1}{3\mathrm{bc}}. \end{aligned} \tag{11}$$

Moreover, messages are required to travel in the x dimension to reach their relative vertical partitions. For example, if $a = 4$ in $a \times b \times c$ network and the source switch is located at the first vertical partition, it takes 1, 2, and 3 hops to reach the second, third, and fourth vertical partitions, respectively. This value should be considered when measuring the MML value:

$$\mathrm{MML_a} = \frac{1}{\mathrm{a}} \sum_{\mathrm{j=1}}^{\mathrm{j=a}} \frac{1}{\mathrm{a}} \left(\sum_{\mathrm{i=1}}^{\mathrm{i=j-1}} \mathrm{i} + \sum_{\mathrm{i=j+1}}^{\mathrm{i=a}} (\mathrm{i-j}) \right) = \frac{\mathrm{a}^2 - 1}{3\mathrm{a}}. \tag{12}$$

Finally, the MML value for the whole network is given by

$$\begin{aligned} \mathrm{MML_{VBP}} &= \mathrm{MML_a} + \mathrm{MML_{bc}} = \frac{\mathrm{a}^2 - 1}{3\mathrm{a}} + \frac{(\mathrm{bc})^2 - 1}{3\mathrm{bc}} \\ &= \frac{\mathrm{a}^2\mathrm{bc} + \mathrm{ab}^2\mathrm{c}^2 - \mathrm{bc} - \mathrm{a}}{3\mathrm{abc}}. \end{aligned} \tag{13}$$

From another point of view, the network can be viewed as a 2D network ($a \times b'$) where $b' = b \times c$. The dimension-order routing can be utilized for messages, and thus, by using (1) in a 2D network (when $c = 1$) the average multicast latency can be measured by

$$\mathrm{MML_{VBP}} = \frac{\mathrm{a}^2\mathrm{b}' + \mathrm{ab}'^2 - \mathrm{a} - \mathrm{b}'}{3\mathrm{ab}'} = \frac{\mathrm{a}^2\mathrm{bc} + \mathrm{ab}^2\mathrm{c}^2 - \mathrm{bc} - \mathrm{a}}{3\mathrm{abc}}.$$

In the VBP method, the network is divided into several vertical partitions according to the value a in $a \times b \times c$ network. Thereby, the following formula is used for computing the MxML value in the network:

$$\mathrm{MxML_{VBP}} = \frac{2}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{i=n/2}} \left(\left\lceil \frac{\mathrm{n-j}}{\mathrm{a}} \right\rceil + \frac{\mathrm{a}^2 - 1}{3\mathrm{a}} \right). \tag{14}$$

The number of partitions in the VBP method depends on the location of switches that results in different startup messages. The switches in the first row of the first layer and the last row of the last layer divide the network into 4, 5, 6, and 7 partitions, while the other switches divide the network into eight partitions. As a result, the average number of startup messages for the VBP method in $a \times b \times c$ network is

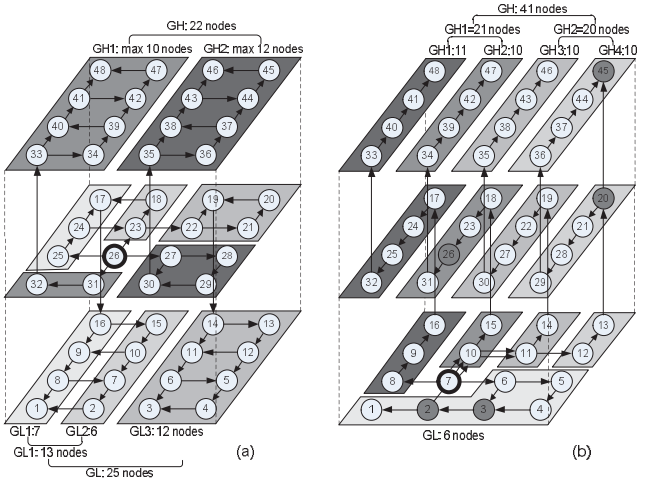


Fig. 4. RP when the source switch is at (a) switch 26, (b) switch 7.

$$\mathrm{SM_{VBP}} = \frac{(3\mathrm{a}^2 - \mathrm{a}) + ((\mathrm{abc} - 2\mathrm{a})(2\mathrm{a}))}{\mathrm{abc}} = \frac{2\mathrm{a}^2\mathrm{bc} - \mathrm{a}^2 - \mathrm{a}}{\mathrm{abc}}. \tag{15}$$

3.2.3 Recursive Partitioning

The objective of the recursive partitioning method is to optimize the number of switches that can be included in a partition and achieve parallelism. In this method, the network is recursively partitioned until each partition contains k switches. In the worst case, the network is partitioned into 2a vertical partitions like in the VBP method. We have considered the value k as a reference value indicating the number of switches in each partition of the VBP method, i.e., $(k = bc)$ in $a \times b \times c$ network. An example of the RP method is illustrated in Fig. 4a where a multicast message is generated at the source switch 26. The required steps of the RP method can be expressed as follows.

Step1: The value k is set to 12 switches in a $4 \times 4 \times 3$ network.

Step2: The network is divided into two partitions using the TBP method. Fig. 2a shows two formed partitions when the source switch is located at switch 26.

Step3: If the number of switches in a partition exceeds the reference value k, the partition is divided into two new partitions. This step is repeated until all partitions in the network cover at most k switches. Following the example of Fig. 2a, 22 switches are covered by the high channel subnetwork which is greater than $k = 12$. The high channel subnetwork needs to be further divided into two new partitions (GH1 and GH2 as shown in Fig. 4a). The GH1 and GH2 partitions contain 10 and 12 switches, respectively. Since both numbers are less than or equal to $k = 12$, no further partitioning is needed for the high channel subnetwork. The same partitioning technique is applied to the low channel subnetwork.

Fig. 4b shows another example of the RP method where the multicast message is $m = (7, \{2, 3, 20, 26, 45\})$. In this

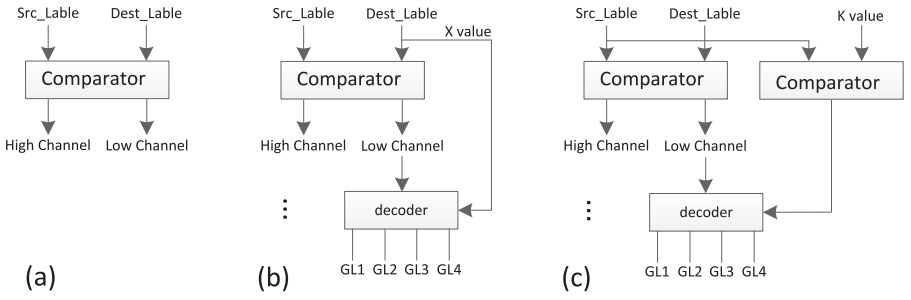


Fig. 5. The microarchitecture of (a) TBP, (b) VBP, and (c) RP methods.

example three messages are formed and their paths are *{7,10,11,12,13,**20,45**},{7,**26**}*, and *{7,6,**3,2**}* as the maximum latency is six hops.

In brief, this scheme has a similar performance in avoiding long paths since the VBP method while it provides better parallelism as the number of switches is comparable among partitions. By considering the RP method, the creation of balanced partitions is less dependent of the source switch position, and thus it avoids long paths in the network and increases parallelism while keeping the number of startup messages relatively low.

The startup latency and network latency for multicast messages. In the RP method, each subnetwork is recursively partitioned until all partitions cover around k switches, where $k = bc$. The next set of formulas is concerned only the high channel subnetwork while the low channel subnetwork has similar formulas. According to this assumption, if the high channel subnetwork covers x switches where $x > k$, it is divided into two new partitions. Each of the formed partitions might still cover more than k switches ($x > k$). Thereby, the partition is further divided into two new partitions. In other words, the MML formula is recursively called until all partitions cover at most k switches. Finally, the average multicast latency is computed when the number of switches x in a partition become less than or equal to the value of k:

$$\mathrm{MML_x} = \begin{cases} \dfrac{\mathrm{MML_{\frac{x}{2}}} + \mathrm{MML_{\frac{x}{2}}}}{2} & \text{where } \mathrm{x} > \mathrm{k} \\ \dfrac{1}{\mathrm{x}} \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{x}} \mathrm{i} = \dfrac{\mathrm{x}+1}{2} & \text{where } 0 < \mathrm{x} \leq \mathrm{k} \\ 0 & \text{where } \mathrm{x} = 0. \end{cases} \tag{16}$$

Similar to (12), in order to deliver messages from the source switch to different partitions, average multicast latency in the x dimension should be taken into account. Finally, the MML for the RP method is given by

$$\begin{aligned} \mathrm{MML_{RP}} &= \frac{1}{2\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} \left(\mathrm{MML}^{\mathrm{low}}_{(\mathrm{j}-1)} + \mathrm{MML}^{\mathrm{high}}_{(\mathrm{n}-(\mathrm{j}+1))} \right) + \mathrm{MML_a} \\ &= \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{n}} \mathrm{MML}_{(\mathrm{i}-1)} + \mathrm{MML_a}. \end{aligned} \tag{17}$$

For measuring MxML, the number of switches in the biggest partition should be identified. To do this, we first find the MxML value for the high and low channel subnetworks and then determine the number of switches in the biggest partition of the network as

$$\mathrm{MxML} = \mathrm{Max}(\mathrm{MxML^{High}}, \mathrm{MxML^{Low}}) \quad \text{where}$$

$$\mathrm{MxML_x^{high}} \text{or } \mathrm{MxML_x^{Low}} \begin{cases} \mathrm{Max}\left(\mathrm{MxML}_{\lceil \frac{\mathrm{x}}{2} \rceil}, \mathrm{MxML}_{\lfloor \frac{\mathrm{x}}{2} \rfloor}\right) & \\ \qquad \text{where } \mathrm{x} > \mathrm{k} & \\ \dfrac{1}{\mathrm{x}} \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{x}} i = \dfrac{\mathrm{x}+1}{2} & \\ \qquad \text{where } 0 < \mathrm{x} \leq k & \\ 0 \qquad \text{where } \mathrm{x} = 0. & \end{cases} \tag{18}$$

To compute the MxML value, the following formula is utilized:

$$\mathrm{MxML_{RP}} = \frac{2}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{n}/2} (\mathrm{MxML} + \mathrm{MML_A}). \tag{19}$$

In the case that $x \leq k$, the number of startup messages is equal to 1. However, when $x > k$, the partition needs to be divided into two new partitions and the SM equation is called for every newly formed partition

$$\mathrm{SM_x} = \begin{cases} \mathrm{SM}_{\lceil \frac{\mathrm{x}}{2} \rceil} + \mathrm{SM}_{\lfloor \frac{\mathrm{x}}{2} \rfloor} & \text{when } \mathrm{x} > \mathrm{k} \\ 1 & \text{when } \mathrm{x} \leq \mathrm{k}. \end{cases} \tag{20}$$

3.3 Hardware Implementation

The microarchitectures of the TBP, VBP, and RP methods are illustrated in Figs. 5a, 5b, and 5c, respectively. In all three methods, the source label is compared (using a comparator) with the destinations labels, so that destinations are divided into high and low channel subnetworks (Fig. 5a). In the VBP method, by decoding the value x of the destination address, each destination is placed in one partition as shown in Fig. 5b. In the RP method, however, the number of switches in the high (or low) channel subnetwork is compared with the reference value k. The result of this comparison determines the required number of partitions such that each can cover about k switches. In the next step, destinations are divided into different partitions (Fig. 5c). All procedures are performed in the packetizer unit of the network interface and repeated for every destination in the destination set [34]. Finally, each nonempty register is used in the header of a message. Notice that for encoding the addresses in the message header, we have utilized the bit string scheme [3], where each bit corresponds to a switch in a network. For a set of destinations, the corresponding bits in the bit-string become one.

```
Algorithm: Minimal Adaptive Routing (MAR_3D)
Inputs:    current switch label, destination switch label,
neighboring switches Labels
Begin
                ----------STEP 1----------
   X_dir = East when (x_c<x_d) else West;
   Y_dir = North when (y_c<y_d) else South;
   Z_dir = High when (z_c<z_d) else Low;
Process     ----------STEP 2----------
   Begin
      If ((Label(CurrentSwitch) = Label(DestSwitch)) then
           Select Local;
      Elsif ((Label(CurrentSwitch) < Label(DestSwitch)) then
----------High Channel Subnetwork---------
          If (Label(CurrentSwitch) < Label(Neighbor(X_dir))) and
            (Label(Neighbor(X_dir)) < Label(DestSwitch)) then
               First Choice -> Neighbor(X_dir)
          End if;
          If (Label(CurrentSwitch) < Label(Neighbor(Y_dir))) and
            (Label(Neighbor(Y_dir)) < Label(DestSwitch)) then
               Second Choice -> Neighbor(Y_dir)
          End if;
          If (Label(CurrentSwitch) < Label(Neighbor(Z_dir))) and
            (Label(Neighbor(Z_dir)) < Label(DestSwitch)) then
               Third Choice -> Neighbor(Z_dir)
          End if;
      Elsif ((Label(CurrentSwitch) >Label(DestSwitch)) then
----------Low Channel Subnetwork---------
          If (Label(CurrentSwitch) >Label(Neighbor(X_dir))) and
            (Label(Neighbor(X_dir)) > Label(DestSwitch)) then
               First Choice -> Neighbor(X_dir)
          End if;
          If (Label(CurrentSwitch) > Label(Neighbor(Y_dir))) and
            (Label(Neighbor(Y_dir)) > Label(DestSwitch)) then
               Second Choice -> Neighbor(Y_dir)
          End if;
          If (Label(CurrentSwitch) >Label(Neighbor(Z_dir))) and
            (Label(Neighbor(Z_dir)) > Label(DestSwitch)) then
               Third Choice -> Neighbor(Z_dir)
          End if;
    End If;
End Process;
```

Fig. 6. The pseudocode of the MAR algorithm.

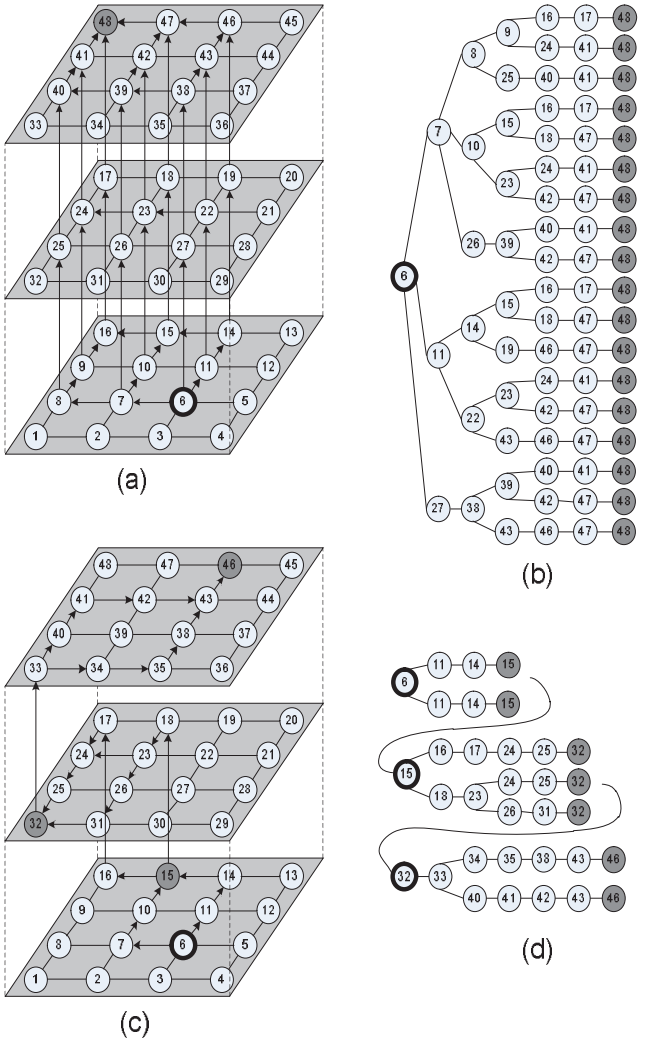


Fig. 7. Example of the MAR algorithm for unicast and multicast messages.

4 Minimal Adaptive Routing (MAR)

In the previous section, we provided different partitioning methods. All of them require a routing algorithm capable of forwarding all the messages to their sets of destinations. In this section, we present a minimal and adaptive routing algorithm based on the Hamiltonian path. Using MAR, unicast and multicast messages can be adaptively routed inside the network. The MAR algorithm is implemented at switches and can be described in three steps as follows:

Step1: it determines the neighbors of current switch u that can be used to move a message closer to its destination d. The pseudocode for Step1 is shown in Fig. 6.

Step2: due to the fact that in the Hamiltonian path all switches are visited in ascending order (in the high channel subnetwork) or descending order (in the low channel subnetwork), all of the selected neighbors in Step1 do not necessarily satisfy the ordering constraint. Therefore, if the labels of the selected neighbors (in Step1) are between the label of switch u and destination d, it/they can be selected as the next hop. The pseudocode for Step2 is shown in Fig. 6.

Step3: since the MAR algorithm provides several choices at each switch, the goal of Step3 is to route a message through the less congested neighboring switch. In the case where the message can be forwarded through multiple neighboring switches, the stress values of the input buffers in the selected neighbors are checked and then the message is sent to the neighbor with the smallest stress value. An example of the MAR algorithm is illustrated in Fig. 7a. According to the algorithm, in the first step the neighbors are chosen in a manner that gets the message closer to its destination, i.e., $p = \{7, 11, 27\}$. At the second step, the selected neighbors (in Step1) are checked to determine whether they are in the Hamiltonian path or not. Since the labels of the three selected neighbors are between the labels of the current switch $(u = 6)$ and destination switch $(d = 48)$, the message can be routed via all of them. Suppose that the neighbor $p = 11$ has a lower stress value than the other neighbors, so the algorithm chooses this neighbor to forward the message. If we continue with the switch $u = 11$, this switch has three neighbors belonging to the minimal paths, i.e., $p = \{10, 14, 22\}$. However, only two of them $(p = \{14, 22\})$ have the labels greater than the label of the current switch $(u = 11)$ and lower than the label of the destination switch $(d = 48)$. Finally, according to the stress values of the input buffers in the corresponding direction, one of them is

TABLE 1
Unicast Latency, Startup Messages for Different Number of Destinations, Message Lengths, and Injection Rates

Method	1st Size	2nd UL (hop)	3rd SM	4th SM 8 D/M 1 F/M	5th SL 8 D/M 5 F/M 1% R 1st M	6th SL 8 D/M 5 F/M 10% R 100th M	7th SM 16 D/M 1 F/M	8th SL 16 D/M 10 F/M 1% R 1st M	9th SL 16 D/M 10 F/M, 10% R 100th M
TBP	4×4×4	3,75	2	2	5	5	2	10	10
TBP	8×8×8	7,88	2	2	5	5	2	10	10
VBP	4×4×4	3,75	8	5	20	20	7	60	60
VBP	8×8×8	7,88	16	6	25	25	11	100	100+990
RP	4×4×4	3,75	5	4	15	15	5	40	40
RP	8×8×8	7,88	10	5	20	20	8	70	70

UL: Unicast Latency; SM: Startup Messages; SL: Startup Latency; D/M: Destination per Message; F/M: Flit per Message; R: Rate.

selected as the next hop. The algorithm is repeated for the rest of the switches until the message reaches the final destination. Fig. 7b shows all possible shortest paths from the source switch $(u = 6)$ to the destination switch $(d = 48)$. It is worth noting that the stress value is updated whenever a new flit enters or leaves the buffer (flit events: flit_tx or flit_rx). That is, in each flit event, if the number of occupied cells of the input buffer is larger (smaller) than a threshold value, the threshold signal is assigned to one (zero).

The MAR algorithm can be adapted for multicast messages such that alternative paths are used to route a message between the source switch and the first destination and also between successive destinations. An example is shown in Fig. 7c where the source $(u = 6)$ forwards a multicast message toward its destinations $(D = \{15, 32, 46\})$. The MAR algorithm provides a set of alternative paths to send a message from the source switch to the first destination $(d1 = 15)$. Similarly, the message can be adaptively routed between each two destinations. For example, at switch 15, the message can make progress toward destination 32 either by selecting switch 18 in the next layer or switch 16 in the current layer. The MAR algorithm is compatible with all methods supporting the Hamiltonian path in 2D or 3D NoCs. Therefore, all the TBP, VBP, and RP methods can utilize the MAR algorithm for both unicast and multicast messages. Fig. 7d shows all possible shortest paths from the source switch $(u = 6)$ to the destinations 15, 32, and 46.

To show that the proposed algorithm is deadlock free, we need to prove that the channel dependency graph (CDG) is acyclic [33]. To close a cycle in the high channel subnetwork, a message may require requesting a channel that forwards the message to a lower labeled switch, which is not allowed by the MAR algorithm. The same applies for the low channel subnetwork. Since both in unicast and multicast traffic, messages are routed only in ascending and descending order, the MAR algorithm is deadlock free. However, in multicast traffic there is a possibility of deadlock in the consumption channels [33]. This happens when a message should be delivered both to the local node and the output channel (to move toward the next destination). This may cause deadlock if two multicast messages reach the switch and request both channels, but each gets access to only one channel. There is a branch dependency that creates a deadlock situation. This can be solved basically using extra resources between the local core and the corresponding switch to avoid such conflict. In our case, we implement at each switch two ejection channels.

5 Results and Discussion

5.1 Analytical Results

We analyzed and compared the unicast latency, the startup latency, and the network latency of the TBP, VBP, and RP partitioning methods using analytical models. For this purpose, the previously presented factors (SM, MML, and MxML) are utilized. For each method, we explore the values for two different network sizes along with two different numbers of destinations, injection rates, and message lengths. Finally, the total latency is estimated under different configurations and methods.

5.1.1 Startup Latency

We developed formulas to extract the number of startup messages of the TBP, VBP, and RP methods. However, the startup latency not only depends on the SM value but also it is affected by the message length, injection rate, and the number of destinations per multicast message.

The impact of the number of destinations on the startup latency. We computed the upper bound value of SM for the TBP, VBP, and RP methods by assuming that there is one message per partition. The third column of Table 1 shows the maximum number of startup messages in the TBP, VBP, and RP methods. However, in reality, the number of messages may be smaller than the number of partitions (e.g., when the number of destinations is lower than the sets or destinations are not evenly distributed among sets). We have assumed uniform distribution and used conditional probabilities to find out the probability that a partition has received a message when there are 8 or 16 destinations per message. Based on this evaluation, the fourth and seventh columns in Table 1 are filled. For example, when there are eight partitions and eight destinations per message, on average, five partitions include at least a destination and three partitions are empty, thus the average startup messages is five. As the number of destinations per message increases (e.g., from 8 to 16 destinations), with a high probability there is at least one destination per partition. In

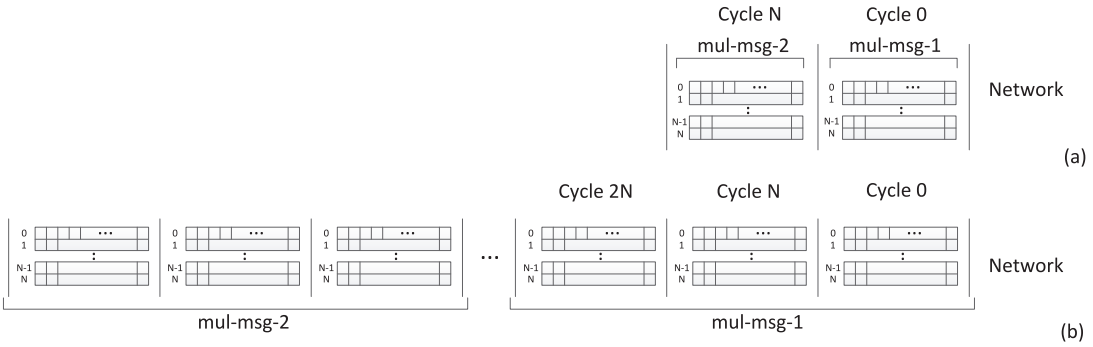


Fig. 8. The impact of message length on the startup latency.

this case, the startup messages almost reach the upper bound values. According to the values in Table 1, the RP method offers a lower startup messages than the VBP method since partitions are merged together.

The average unicast latency for different network sizes is listed in the second column. As already mentioned, the unicast latencies of different methods are similar. This is because of the fact that unicast messages are routed similarly in the network using the TBP, VBP, and RP methods. Obviously, the unicast latency is increased as the network scales up.

The impact of message length on the startup latency. To show the impact of the message length on the startup latency, let us assume that a multicast message carries all destinations' addresses, and thus only one message is sent to the network. In the TBP method and when there is no contention in the network, the first flit of the message 1 (mul-msg1) enters the network at cycle 0 while the message 2 (mul-msg2) can start sending its first flit at cycle N, where N is the number of flits per message (Fig. 8a). By partitioning the network in the VBP and TP methods, the destinations are distributed among several sets. In this case, multiple copies of the message 1 (with different sets of destinations) are injected into the network at cycles $0, N, 2N$, etc. (Fig. 8b). The message 2 can deliver its first flit as soon as all copies of the message 1 are delivered into the network. We compute the startup latency by considering the average message length as follows:

$$\mathrm{SL_A} = (\text{startup messages} - 1) * (\text{flits per message}).$$

In Table 1, the fourth and fifth columns indicate the differences between the startup latencies when the message size increases from one to five flits. Similarly, the seventh and eighth columns show the startup latencies by changing the message size from one to ten flits. The values show an increased in the startup latencies when the message size increases. In all configurations, the TBP method has the lowest startup latency, and then the RP and VBP methods, respectively.

The impact of the injection rate on the startup latency. In a low injection rate, the message 2 is probably generated by the core when all messages of the message 1 have already sent to the network. However, in a case of high injection rate, the message 2 is ready to be sent to the network while the messages of message 1 are still in the queue and have not completely delivered to the network. Therefore, if the number of cycles required for delivering all the messages of a multicast message is larger than $(100 - rate\%)$, the following formula is obtained: (Latency is cumulative, with each additional generated message)

$$\mathrm{SL_B} = \mathrm{SL_A} + (\text{total number of generated messages} - 1) * (\mathrm{SL_A} - (100 - \text{rate}\%)).$$

Table 1 also includes the results when the injection rate takes into consideration. The values are obtained based on two injection rates, 1 and 10 percent. As can be seen in the fifth and sixth columns (or eighth and ninth columns), in most cases, the startup latencies do not change as the message 1 has delivered all its messages before the message 2 is generated. However, in a high injection rate (i.e., 10 percent), the time required to send startup messages may exceed $100 - 10 = 90$ cycles. As shown in the ninth column, in one case, it takes more than 90 cycles to deliver startup messages completely to the network. Indeed, the newly generated messages experience considerably larger delays to send their first flit into the network. The values in the sixth and ninth columns are computed for the 100th message, while in fifth and eighth columns are measured for the first message.

5.1.2 Network Latency

Using analytical formulas, we have estimated the MxML and MML values for TBP, VBP, and RP methods in $4 \times 4 \times 4$ and $8 \times 8 \times 8$ networks. Since MxML and MML reveal the number of hops, to estimate the network latency, the switch delay should be taken into consideration. By assuming 3-stage pipeline architecture, the network latency is computed by multiplying the number of hops and a factor of three. On the other hand, as the injection rate and contention increases, per-hop delay is increased. We assume that in a 10 percent injection rate, on average, the latency is six cycles per hop. According to this assumption, we estimate the total latency using the following formula:

$$\text{Total Latency} = \begin{cases} \text{MML} * 3 + \text{SM} & \text{with } 1\% \text{ injection rate} \\ \text{MML} * 6 + \text{SM} & \text{with } 10\% \text{ injection rate.} \end{cases}$$

The values in second and third column of Table 2 indicate that MxML and MML of the TBP method are considerably larger than those of values in the VBP and RP methods. The VBP method can reduce the MML value significantly at a cost of more startup messages. Fourth, fifth, sixth, and seventh columns show the total latency values when the startup latency takes into consideration. Since, the high number of startup messages in the VBP method may result in a time overlapping of different

TABLE 2
MML, MXML, and Total Latency in TBP, VBP, and RP Methods

Method	1st Size	2nd MxML	3rd MML	4th MML*3+SL 5flits,8dests, 1%rate, 1th message	5th MML*6+SL 5flits,8dest, 10% rate, 100th message	6th MML*3+SL 10flits,16dests, 1%rate, 1th message	7th MML*6+SL 10flits,16dest, 10% rate, 100th message
TBP	4×4×4	48	21	68	131	73	136
TBP	8×8×8	384	171	518	1031	523	1036
VBP	4×4×4	14	6	38	56	78	96
VBP	8×8×8	51	24	97	169	172	1234
RP	4×4×4	15	7	36	57	61	82
RP	8×8×8	59	26	98	176	148	226

messages, as can be seen in the last column, in some cases the VBP method even behave worse than the TBP method.

5.2 Simulation Results

To assess the efficiency of the proposed partitioning methods in experiment, we have developed a cycle-accurate NoC simulator based on wormhole switching in 3D mesh configuration. The simulator inputs include the array size, the routing algorithm, the link width, the buffer size, and the traffic type. The on-chip network, considered for experiment is formed by a typical wormhole-switching structure including input buffers, a routing unit, a switch allocator, and a crossbar. Each switch has seven input/output ports, a natural extension from a 5-port 2D switch by adding two ports to make connections to the upper and lower layers [28], [36]. There are some other types of 3D switches such as the hybrid switch [5], [36], [37] and MIRA [38], however, since switch efficiency is out of the goals of this paper, we have chosen a simple 7-port switch in our simulation. The arbitration scheme of the switch allocator in the typical switch structure is round robin. The data width and the frequency were set to 64 bits and 1 GHz, respectively, and each input channel has a buffer size of five flits with the congestion threshold at 80 percent of the total buffer capacity. This congestion threshold is utilized by the presented MAR algorithm to choose the less congested path if there would be any alternative path(s). The experiments were performed on a 48-switch ($4 \times 4 \times 3$) 3D stacked architecture with a constant message size of five flits. For the performance metric, we used the multicast latency defined as the number of cycles between the initiation of a multicast message operation, including preparation and startup latency, and the time when the tail of the multicast message reaches all the destinations. For each load value, the result of message latency is averaged over 80,000 messages after a warm-up session of 20,000 arrived messages.

5.2.1 Performance Evaluation

Multicast traffic profile. The first set of simulations was performed for a random traffic profile. A uniform distribution is used to construct the destination set of each multicast message [18]. The number of destinations has been set to 8 or 16. The average communication delay as a function of the average message injection rate has been shown in Fig. 9. As observed from the results, the RP method meets lower delay than the TBP and VBP methods. The foremost reason for this performance gain is due to the efficiency of the RP method which not only reduces the number of hops for multicast messages but also the number of startup messages. In fact, TBP suffers from long paths while the performance of VBP degrades due to a large number of startup messages. Adaptive routing algorithms obtain better performance in congested networks due to using alternative routing paths [23]. In Fig. 10, ARP (Adaptive RP, utilizing MAR in RP), and AVBP (Adaptive VBP, utilizing MAR in VBP) are the adaptive models of the RP and VBP methods, respectively. As illustrated in this figure, adaptive routings become more advantageous when the injection rate increases.

Unicast and multicast (mixed) traffic profile. In this set of simulations, we used a mixture of unicast and multicast traffic, where 70 percent of injected messages are unicast messages and the remaining 30 percent are multicast

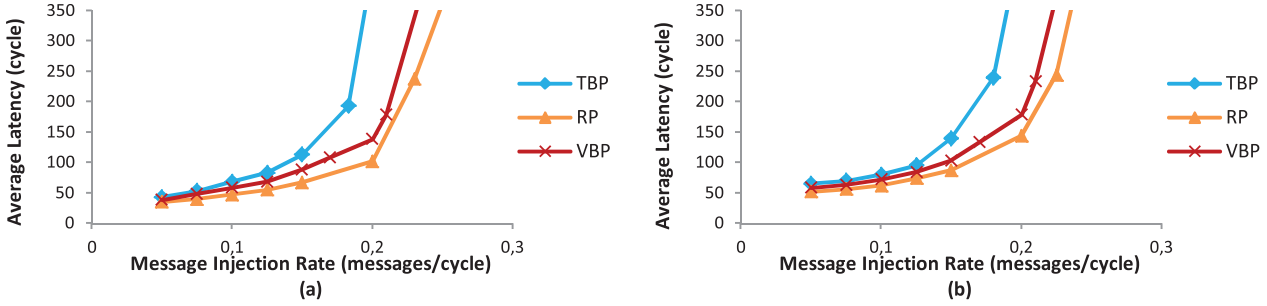


Fig. 9. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using deterministic routing with (a) eight destinations, (b) 16 destinations.

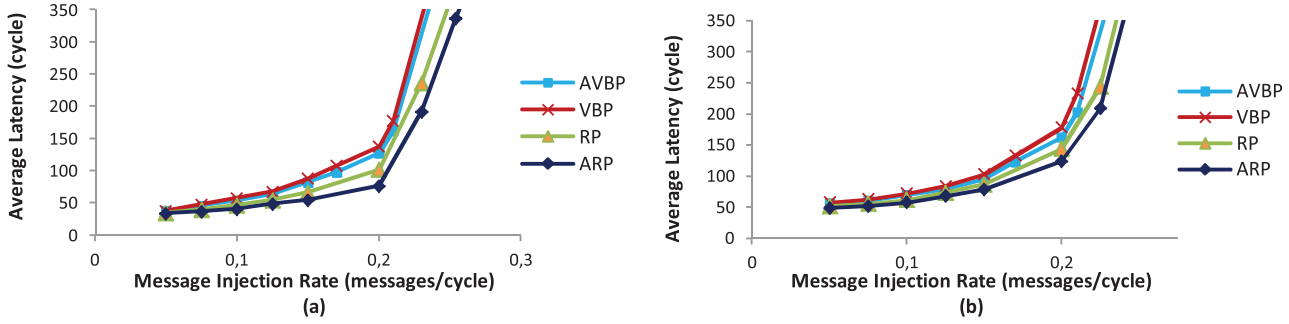


Fig. 10. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using adaptive routing with (a) eight destinations, (b) 16 destinations.

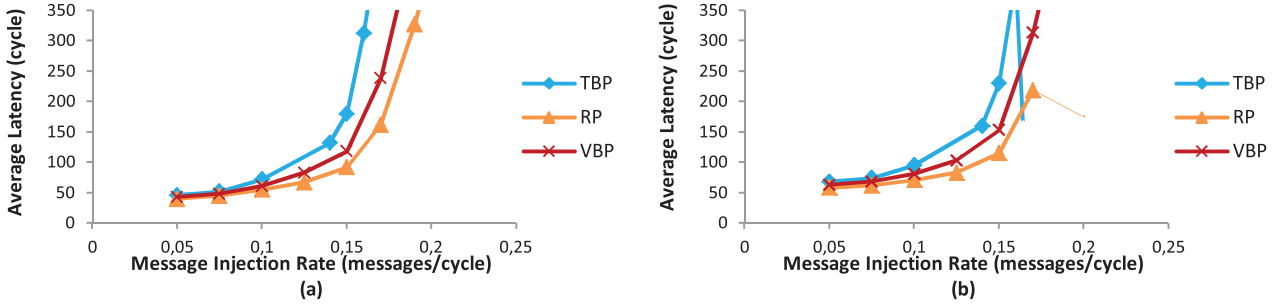


Fig. 11. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using deterministic routing with (a) eight destinations, (b) 16 destinations under mixed traffic (30 percent multicast and 70 percent unicast); unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and $h = 10\%$.

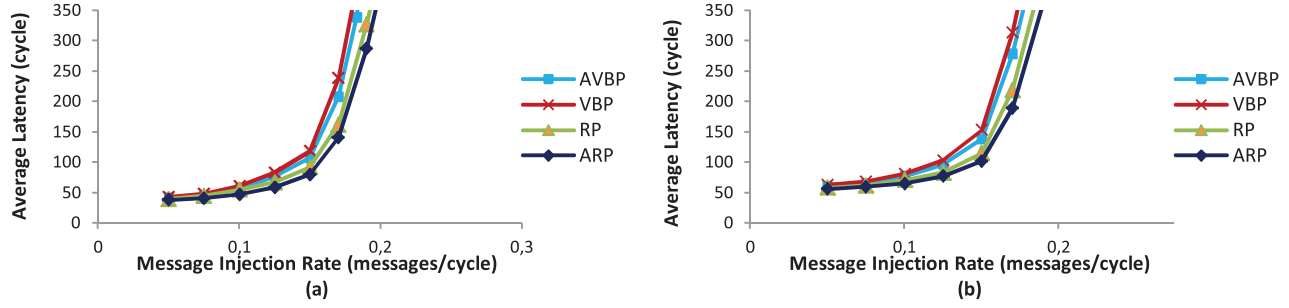


Fig. 12. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using adaptive routing with (a) eight destinations, (b) 16 destinations under mixed traffic (30 percent multicast and 70 percent unicast); unicast traffic is based on the hotspot traffic model with a single hotspot switch (2,2,2), and $h = 10\%$.

messages. Hotspot and transpose traffic model profiles [39] have been taken into account for unicast traffic generation. Under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In the hotspot traffic model, given a hotspot percentage of h, a newly generated message is directed to each hotspot switch with an additional h percent probability. We simulate hotspot traffic with a single hotspot switch. The hotspot switch is chosen to be the switch (2,2,2) in the $4 \times 4 \times 3$ mesh network. Fig. 11 shows the performance with $h = 10\%$.

Under the transpose traffic pattern, the source switch positioned at (x, y, z) sends messages to the destination switch $(a-1-x, b-1-y, c-1-z)$ for all $x\epsilon\{0, ..., a-1\}$, $y\epsilon\{0, ..., b-1\}$, $z\epsilon\{0, ..., c-1\}$, in $a \times b \times c$ mesh network. As illustrated in Figs. 11 and 13, the RP method outperforms the two other partitioning methods under both traffic profiles when using a deterministic routing algorithm. This improvement is achieved through using optimized partitions formed by the RP method. Moreover, Figs. 12 and 14 show the average latency when utilizing the MAR routing algorithm. Based on the presented partitioning methods, the adaptive routing reduces the average latency in comparison with the deterministic routing.

Application traffic profile. The GEMS full system simulator [17] is used as our simulation platform coupled with a cycle-accurate 3D NoC model. In order to know the real impact of the presented methods, we used traces from some application benchmark suites selected from SPLASH-2 [12], and PARSEC [13], [14]. Simulations are run on the Solaris 9 operating system based on SPARC instruction architecture. The adopted mapping strategy used in Solaris 9 is arbitrary mapping. Table 3 summarizes our full system configuration where the cache coherence protocol is token-based MOESI and access latency to the $L2$ cache is derived from the CACTI [40]. We form a 64-node on-chip network $(4 \times 4 \times 4)$ that four layers are stacked on top of each other, i.e., out of the 64 nodes, 16 nodes are processors and other 48 nodes are $L2$ caches. $L2$ caches are distributed in the bottom three layers, while all the processors are placed in the top layer close to a heat sink so that the best heat dissipation capability is achieved [38], [41]. For the

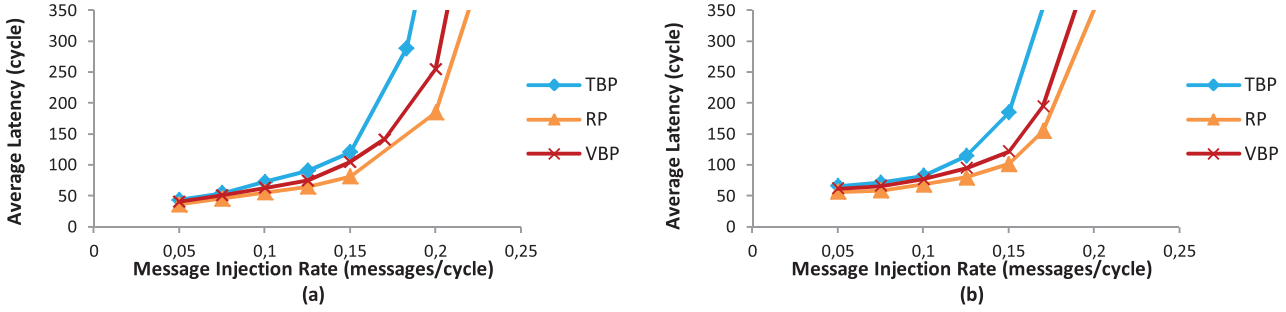


Fig. 13. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using deterministic routing with (a) eight destinations, (b) 16 destinations under mixed traffic (30 percent multicast and 70 percent unicast); unicast traffic is based on the transpose traffic model.

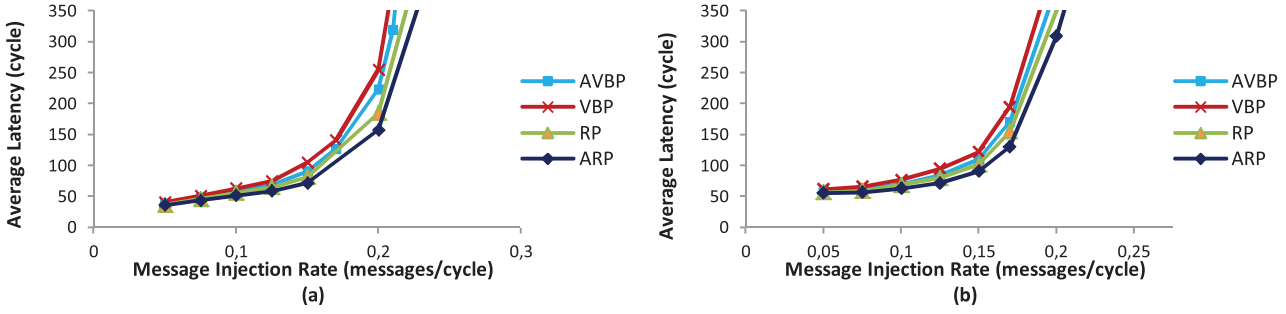


Fig. 14. Performance under different loads in $4 \times 4 \times 3$ 3D mesh using adaptive routing with (a) eight destinations, (b) 16 destinations under mixed traffic (30 percent multicast and 70 percent unicast); unicast traffic is based on the transpose traffic model.

TABLE 3
System Configuration Parameters

Processor Configuration	
Instruction set	SPARC, 16 processors
L1 cache	16KB. 4-way associative, 64-bit line, 3-cycle access time
L2 cache	Shared, distributed in 3 layers, unified, 48MB (48 banks, each 1MB). 64-bitline, 6-Clock
Cache coherence protocol	MESI, Token-based MOESI
Cache hierarchy	SNUCA
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Benchmarks	SPLASH-2, PARSEC
Network configuration	
switch scheme	3D mesh with wormhole
Flit size	64 bits
Workloads	
SPLASH-2	Barnes, Cholesky, FFT, LU, Ocean, Radix, Raytrace, Water-Nsq
PARSEC	x264

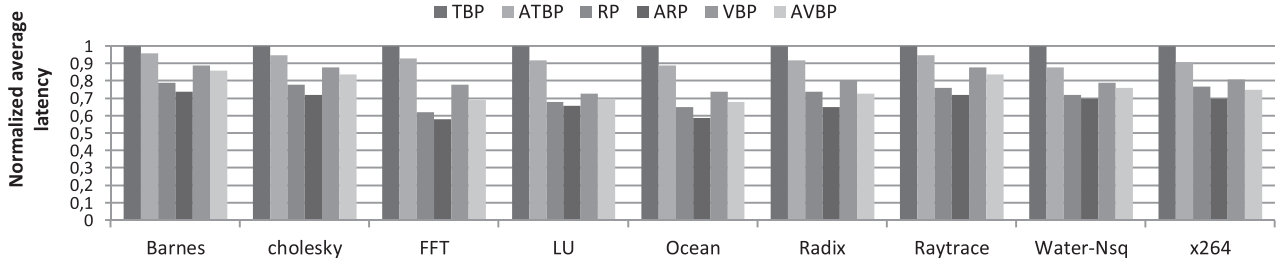


Fig. 15. Performance under different application benchmarks normalized to TBP.

processors, we assume a core similar to Sun Niagara and use SPARC ISA [42]. The memory hierarchy implemented is governed by a two-level directory cache coherence protocol. Each processor has a private write-back $L1$ cache (split $L1$ I and D cache, 64 KB, 2-way, 3-cycle access). The $L2$ cache is shared among all processors and split into banks (48 banks, 1 MB each for a total of 48 MB, 6-cycle bank access), connected via on-chip switches. The $L1/L2$ block size is

TABLE 4
Performance Gain of ARP over Other Presented Schemes

	TBP	ATBP	RP	VBP	AVBP	
Barnes	26%	23%	6%	16%	14%	
Cholesky	28%	24%	7%	18%	14%	
FFT	42%	37%	6%	25%	15%	
LU	34%	28%	3%	9%	6%	
Ocean	41%	33%	9%	20%	13%	
Radix	35%	29%	12%	18%	10%	
Raytrace	28%	24%	5%	17%	14%	
Water-Nsq	30%	20%	3%	11%	7%	
x264	30%	23%	9%	13%	6%	**Overall**
Avg.	32%	27%	7%	17%	11%	19%

64 B. The simulated memory hierarchy mimics SNUCA [43] while the off-chip memory is a 4 GB DRAM with a *260*-cycle access time. The simulator produces, as output, the communication latency for cache access. Fig. 15 shows the average network latency of the real workload traces collected from the aforementioned system configurations, normalized to TBP. However, using the adaptive routing scheme, MAR, diminishes the average delay of each partitioning method significantly under all benchmarks. That is, adaptive routing has an opportunity to improve performance. For instance, under the *fft* application, the performance gain of using MAR in TBP, RP, and VBP is about (ATBP/TBP) 7 percent, (AVBP/VBP) 11.5 percent, and (ARP/RP) 6 percent. We can see that ARP consistently reduces the average network latency across all tested benchmarks. Table 4 lists the performance gains of ARP over TBP, ATBP, RP, VBP, and AVBP where the overall performance gain is about 19 percent.

5.2.2 Hardware Overhead

The presented partitioning methods have been implemented in network interfaces, thereby, to estimate the hardware cost of the proposed methods, the network area of each partitioning scheme, including switches and network interfaces, with the aforementioned configuration were synthesized by Synopsys D.C. using the UMC 90 nm technology with an operating point of 1 GHz and supply voltage of 1 V. We performed place-and-route, using Cadence Encounter, to have precise power and area estimations. Depending on the technology and manufacturing process, the pitches of TSVs can range from 1 to 10 μm square [44]. In this work, the pad size for TSVs is assumed to be $5\ \mu\mathrm{m}^2$ with pitch of around $8\ \mu\mathrm{m}^2$. The area of two-unidirectional vertical channels, 2D switch, and 3D switch are 0.01, 0.18, and $0.23\ \mathrm{mm}^2$, respectively, by considering the link width of 64 bits. Therefore, the overhead of adding TSVs in a 3D switch is less than 4 percent. Different numbers of registers were employed for TBP (the base method), VBP, and RP methods to implement their partitioning mechanisms in network interfaces, leading to different values of area overhead. Comparing the area cost of the TBP with VBP and RP schemes indicates 5 and 6 percent additional overhead, respectively. All partitioning methods use the same routing unit, and thus the differences in area overhead values are related to the implementation of different methods in the network interfaces. It is worth mentioning that the area overhead of the network interface unit alone in the TBP method is about $0.0419\ \mathrm{mm}^2$. The proposed adaptive routing unit (MAR) imposes less than 0.5 percent overhead on a switch in each method and it is independent of the network size.

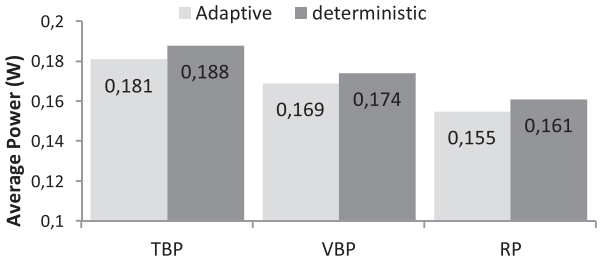


Fig. 16. Average power dissipation results in $4 \times 4 \times 3$ 3D mesh under multicast traffic profile.

5.2.3 Power Consumption

The power consumption of the TBP, VBP, and RP methods were calculated and compared under the multicast traffic model with 16 destinations using the simulator based on the Orion [35] and the equation in [9]. The power values of the network interfaces, computed after the place-and-route in the previous subsection, have been also integrated in the Orion functions. The typical clock of 1 GHz is applied in the aforementioned network ($4 \times 4 \times 3$ 3D mesh network). The results for the average power under multicast traffic are shown in Fig. 16.

The average power values are computed near the saturation point, 0.16 (messages/cycle), under multicast traffic. As the results show, the average power consumption of the RP scheme is 16 and 8 percent less than that of the TBP and VBP schemes, respectively, when using deterministic routing. In fact, this is achieved by smoothly balancing the traffic over the network using efficient balancing scheme which reduces the number of the hot-spots and, hence, lowering the average power.

6 Summary and Conclusion

In this paper, we first presented a set of partitioning methods for 3D mesh-based NoCs along with their analytical models. Among them, the recursive partitioning method achieves higher performance. This method partitions the network recursively until all partitions contain comparable numbers of switches. Experimental results show that the recursive partitioning method reduces the transmission delay and provides a high degree of parallelism compared with the two other methods, TBP and VBP. The paper continued by presenting an adaptive routing algorithm for both unicast and multicast traffic in 3D mesh-based NoCs. The presented algorithm can add adaptivity to the network by taking advantage of the Hamiltonian path strategy without using virtual channels. Using SPLASH-2 and PARSEC benchmarks, the performance gain of the RP method is about 17 and 27 percent, compared with the TBP and VBP methods, respectively, while reducing the power consumption, 12 percent on average.

Acknowledgments

Parts of this paper appeared in the *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, May 2011 [26].

REFERENCES

[1] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. 38th Design Automation Conf.*, pp. 684-689, 2001.

[2] A. Jantsch and H. Tenhunen, *Networks on Chip.* Kluwer, 2003.

[3] J. Duato, S. Yalamanchili, and L.M. Ni, *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann Publishers, 2003.

[4] K. Banerjee, S.J. Souri, P. Kapur, and K.C. Saraswat, "3D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602-633, May 2001.

[5] B.S. Feero and P.P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *IEEE Trans. Computers*, vol. 58, no. 1, pp. 32-45, Jan. 2009.

[6] M. Daneshtalab, M. Ebrahimi, and J. Plosila, "HIBS-Novel Interlayer Bus Structure for Stacked Architectures," *Proc. IEEE Int'l 3D Systems Integration Conf. (3DIC)*, pp. 1-7, Jan. 2012.

[7] H. Matsutani and M. Koibuchi, "Tightly-Coupled Multi-Layer Topologies for 3-D NoCs," *Proc. Int'l Conf. Parallel Processing (ICCP)*, p. 75, 2007.

[8] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, pp. 9-14, 2009.

[9] V.F. Pavlidis and E.G. Friedman, "3-D Topologies for Networks-on-Chip," *IEEE Trans. Very Large Scale Integration Systems*, vol. 15, no. 10, pp. 1081-1090, Sept. 2007.

[10] N.E. Jerger, L.S. Peh, and M. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support," *Proc. 35th Int'l Symp. Computer Architecture (ISCA)*, pp. 229-240, 2008.

[11] P. Abad, V. Puente, and J.A. Gregorio, "MRR: Enabling Fully Adaptive Multicast Routing for CMP Interconnection Networks," *Proc. IEEE 15th Int'l Symp. High Performance Computer Architecture (HPCA)*, pp. 355-366, 2009.

[12] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The Splash-2 Programs: Characterization and Methodological Considerations," *Proc. 22nd Int'l Symp. Computer Architecture*, pp. 24-36, 1995.

[13] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The Parsec Benchmark Suite: Characterization and Architectural Implications," *Proc. 17th Int'l Conf. Parallel Architectures and Compilation Techniques*, pp. 72-81, 2008.

[14] C. Bienia, S. Kumar, and K. Li, "Parsec vs. Splash-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip Multiprocessors," *Proc. IEEE Int'l Symp. Workload Characterization*, pp. 47-56, 2008.

[15] A. Patel and K. Ghose, "Energy-Efficient Mesi Cache Coherence with Pro-Active Snoop Filtering for Multicore Microprocessors" *Proc. 13th Int'l Symp. Low Power Electronics and Design*, pp. 247-252, 2008.

[16] M. Martin, M. Hill, and D. Wood, "Token Coherence: Decoupling Performance and Correctness," *Proc. 30th Ann. Int'l Symp. Computer Architecture*, pp. 182-193, 2003.

[17] M.K. Martin et al., "Multifacet's General Execution Driven Multiprocessor Simulator (GEMS) Toolset," *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92-99, Nov. 2005.

[18] X. Lin and L.M. Ni, "Multicast Communication in Multicomputer Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 10, pp. 1105-1117, Oct. 1993.

[19] E. Carara and F.G. Moraes, "Deadlock-Free Multicast Routing Algorithm for Wormhole-Switched Mesh Networks-on-Chip," *Proc. IEEE CS Ann. Symp.VLSI (ISVLSI)*, pp. 341-346, 2008.

[20] M. Daneshtalab et al., "A Generic Adaptive Path-Based Routing Method for MPSoCs," *Elsevier J. Systems Architecture*, vol. 57, no. 1, pp. 109-120, 2011.

[21] R.V. Boppana, S. Chalasani, and C.S. Raghavendra, "Resource Deadlock and Performance of Wormhole Multicast Routing Algorithms," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 6, pp. 535-549, June 1998.

[22] D. Panda, S. Singal, and R. Kesavan, "Multi Destination Message Passing in Wormhole K-Ary N-Cube Networks with Base Routing Conformed Paths," *IEEE Trans. Parallel and Distributed Systems*, vol. 10, no. 1, pp. 76-96, Jan. 1999.

[23] J. Duato, "On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Theoretical Aspects," *Proc. Second Europe Distributed Memory Computing Conf.*, Apr. 1991.

[24] M. Ebrahimi et al., "HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks," *Proc. ACM/IEEE Sixth Int'l Symp. Networks-on-Chip (NOCS)*, pp. 19-26, May 2012.

[25] M. Dehyadegari et al., "An Adaptive Fuzzy Logic-Based Routing Algorithm for Networks-on-Chip," *Proc. IEEE/NASA-ESA 13th Int'l Conf. Adaptive Hardware and Systems (AHS)*, pp. 208-214, June 2011.

[26] M. Ebrahimi et al., "Exploring Partitioning Methods for 3D Networks-on-Chip Utilizing Adaptive Routing Model," *Proc. ACM/IEEE Fifth Int'l Symp. Networks-on-Chip (NOCS)*, pp. 73-80, May 2011.

[27] L. Wang, H. Kim, and E.J. Kim, "Recursive Partitioning Multicast: A Bandwidth-Efficient routing for Networks-on-Chip," *Proc. Int'l Symp. Networks-on-Chip (NOCS), CA*, pp. 64-73, 2009.

[28] Z. Liu and J. Duato, "Adaptive Unicast and Multicast in 3D Mesh Networks," *Proc. 27th Hawaii Int'l Conf.*, vol. 1, pp. 173-182, 1994.

[29] J. Duato, "A New Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[30] E.O. Amnah and W.L. Zuo, "Hamiltonian Paths for Designing Deadlock-Free Multicasting Wormhole-Routing Algorithms in 3-D Meshes," *J. Applied Sciences*, vol. 7, pp. 3410-3419, 2007.

[31] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer*, vol. 26, no. 2, pp. 62-76, Feb. 1993.

[32] R.S. Ramanujam and B. Lin, "Randomized Partially-Minimal Routing on Three-Dimensional Mesh Networks," *IEEE Computer Architecture Letters*, vol. 7, no. 2, pp. 37-40, July-Dec. 2008.

[33] X. Li, P.K. Mckinley, and L.M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793-804, Aug. 1994.

[34] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, P. Plosila, and H. Tenhunen, "A High-Performance Network Interface Architecture for NoCs Using Reorder Buffer Sharing," *Proc. 18th Euromicro Conf. Parallel, Distributed and Network-Based Processing (PDP)*, pp. 547-550, 2010.

[35] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," *Proc. ACM/IEEE 35th Ann. Int'l Symp. Microarchitecture (MICRO 35)*, pp. 294-305, 2002.

[36] F. Li et al., "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," *Proc. 33rd Ann. Int'l Symp. Computer Architecture (ISCA-33)*, pp. 130-141, 2006.

[37] A.Y. Weldezion, M. Grange, D. Pamunuwa, Z. Lu, A. Jantsch, R. Weerasekera, and H. Tenhunen, "Scalability of Network-on-Chip Communication Architecture for 3D Meshes," *Proc. ACM/IEEE Third Int'l Symp. Networks-on-Chip*, pp. 114-123, 2009.

[38] D. Park et al., "MIRA: A Multi-Layered On-Chip Interconnect Router Architecture," *Proc. 35th Int'l Symp. Computer Architecture (ISCA)*, pp. 251-261, 2008.

[39] G. Chiu, "The Odd-Even Turn Model for Adaptive Routing," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729-738, July 2000.

[40] N. Muralimanohar et al., "Optimizing Nuca Organizations and Wiring Alternatives for Large Caches with Cacti 6.0," *Proc. IEEE/ACM 40th Int'l Symp. Microarchitecture (MICRO)*, pp. 3-14, Dec. 2007.

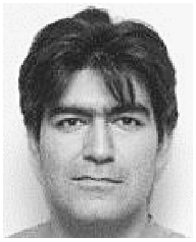
[41] I. Loi and L. Benini, "An Efficient Distributed Memory Interface for Many-Core Platform with 3D Stacked DRAM," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, pp. 99-104, 2010.

[42] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *IEEE Micro*, vol. 25, no. 2, pp. 21-29, Mar./Apr. 2005.

[43] B.M. Beckmann and D.A. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches," *Proc. IEEE/ACM 37th Ann. Int'l Symp. Microarchitecture (MICRO)*, pp. 319-330, 2004.

[44] I. Savidis et al., "Electrical Modeling and Characterization of Through-Silicon Vias (TSVs) for 3D Integrated Circuits," *Microelectronics J.*, vol. 41, no. 1, pp. 9-16, 2010.

Masoumeh Ebrahimi received the PhD degree with honors from the University of Turku, Finland, with over 60 high-level journal and conference paper publications, including three book chapters. Her areas of expertise include interconnection networks, networks-on-chip, 3D integrated systems, and systems-on-chip. During her doctoral studies, she received a 4-year scholarship from the Graduate School in Electronics, Telecommunications and Automation (GETA). In addition, she has received grant awards from the Elisa foundation (2011), Kaute foundation (2013), University of Turku foundation (twice—2011 and 2013), and Nokia foundation (twice—2012 and 2013). She also received a 1-year grant for doing postdoctoral research abroad from the TES foundation. Currently, she holds a VINNOVA Marie-Curie fellowship, conducting her research at KTH University, Sweden, and the University of Turku, Finland, as a senior researcher. She is a member of the IEEE.



Masoud Daneshtalab is an assistant professor in the Department of Information Technology at the University of Turku, Finland. He is an associate editor of the World Research *Journal of Computer Architecture* (*JCA*) and on the editorial boards of *The Scientific World Journal*, *IJDST*, *IJARAS*, *IJERTCS*, and *IJDATICS*. He has served as a guest editor for *Springer Computing* journal, *IET Computers & Digital Techniques*, *ACM Transactions on Embedded Computing Systems* (*ACM TECS*), and *ACM Journal on Emerging Technologies in Computing Systems* (*JETC*), along with Elsevier *Journals of Systems Architecture* (*JSA*), *Microprocessors and Microsystems* (*MICPRO*), *Integration*, and *Computers & Electrical Engineering*. He also coorganizes several special session and workshops, including a regular special session on On-Chip Parallel and Network-Based Systems (OCPNBS) in the Euromicro PDP conference, IEEE International Workshop on High-Performance Interconnection Networks (HPIN) in conjunction with HPCS, and ACM International Workshop on Many-Core Embedded Systems (MES) in conjunction with ISCA, and ACM/IEEE International Workshop Network-on-Chip Architectures (NoCArc) in conjunction with MICRO. He is currently co-supervising three PhD students and his research interests include on/off-chip interconnection networks, many-core embedded systems, embedded operating systems, FPGA and reconfigurable architectures, 3D stacked architectures, machine learning, and cloud data centers. He is a member of the IEEE and has published 1 book, 5 book chapters, and over 100 refereed international journals and conference papers. He is currently a technical program committee member for different IEEE and ACM conferences, including NOCS, DATE, ASPDAC, ESTIMedia, VLSI Design, SOCC, DSD, PDP, EmbeddedCom, ICESS, EUC, NESEA, CASEMANS, NoCArc, MES, HPIN, and JEC-ECC.

Pasi Liljeberg received the MSc and PhD degrees in electronics and information technology from the University of Turku, Turku, Finland, in 1999 and 2005, respectively. He is an associate professor in the Embedded Electronics laboratory and an adjunct professor in embedded computing architectures at the University of Turku, Embedded Computer Systems laboratory. During the period 2007-2009 he held an Academy of Finland researcher position. He is the author of over 150 peer-reviewed publications and has supervised 9 PhD theses. His current research interests include parallel and distributed systems, Internet-of-Things, embedded computing architecture, fault-tolerant and energy-aware system design, 3D multiprocessor system architectures, dynamic power management, cyber physical systems, intelligent network-on-chip communication architectures, and reconfigurable system design. He has established a research group focusing on reliable and fault-tolerant self-timed communication platforms for multiprocessor systems, FastCop project, 2008-2011, Academy of Finland. He is a member of the IEEE.

Juha Plosila is an associate professor in embedded computing and an adjunct professor in digital systems design at the University of Turku (UTU), Department of Information Technology, Finland. He received the PhD degree in electronics and communication technology from UTU in 1999. He is the leader of the Embedded Computer and Electronic Systems (ECES) research unit and a co-leader of the Resilient IT Infrastructures (RITES) research program at the Turku Centre for Computer Science (TUCS). He leads the Embedded Systems master's program at the EIT ICT Labs Master School and is a management committee member of the EU COST Actions IC1103 (MEDIAN: Manufacturable and Dependable Multicore Architectures at Nanoscale) and IC1202 (TACLe: Timing Analysis on Code Level). He is an associate editor of the *International Journal of Embedded and Real-Time Communication Systems* (*IGI Global*). His current research deals with adaptive multiprocessor systems at different abstraction levels. This includes, e.g., specification, development, and verification of self-aware, multi-agent monitoring and control architectures for massively parallel systems, as well as applications of autonomous energy-efficient architectures to new computational challenges in the cyber-physical systems domain. He is a member of the IEEE.

José Flich received the PhD degree in 2001 in computer engineering. He is an associate professor at the Universidad Politécnica de Valencia (UPV), where he leads the research activities related to NoCs. He published more than 100 conference and journal papers, and has served on different conference program committees (ISCA, NOCS, ICPP, IPDPS, HiPC, CAC, CASS, ICPADS, ISCC), as program chair (INA-OCMC, CAC), and as track co-chair (EUROPAR). He has collaborated with different Institutions (Ferrara, Catania, Jonkoping, USC) and companies (AMD, Intel, Sun). Current research activities focus on routing, coherency protocols, and congestion management within NoCs. He has co-invented different routing strategies, reconfiguration, and congestion control mechanisms, some of them with high recognition (RECN and LBDR for on-chip networks). He is a member of the Hipeac-2 NoE. He is coeditor of the book *Designing Network-on-Chip Architectures in the Nanoscale Era*, and is the coordinator of the P7 NaNoC project. He is a member of the IEEE.

Hannu Tenhunen received the PhD degree from Cornell University, Ithaca, New York, in 1985 and since than he has held professor, invited professor, or honorary professor positions in Tampere, Stockholm, Ithaca, Grenoble, Shanghai, Beijing, and Hong Kong. During recent years he has been the director of the Turku Centre of Computer Science and an invited professor at the University of Turku, Finland, where he has established the Computer Systems Laboratory, the leading computer architecture and systems research centre in Finland. His research interests include new computational architectures, dependability issues, on-chip and off-chip communication, and mixed signal and interference issues in complex electronic systems, including 3D integration. He has done more than 600 publications or invited key note talks internationally. He is a member of the IEEE.