J Supercomput (2015) 71:2877–2899
DOI 10.1007/s11227-014-1339-y

In-order delivery approach for 2D and 3D NoCs

Masoud Daneshtalab · Masoumeh Ebrahimi · Sergei Dytckov · Juha Plosila

Published online: 4 December 2014

Abstract In many applications, it is critical to guarantee the in-order delivery of requests from the master cores to the slave cores, so that the requests can be executed in the correct order without requiring buffers. Since in NoCs packets may use different paths and on the other hand traffic congestion varies on different routes, the in-order delivery constraint cannot be met without support. To guarantee the in-order delivery, traditional approaches either use dimension-order routing or employ reordering buffers at network interfaces. Dimension-order routing degrades the performance considerably while the usage of reordering buffers imposes large area overhead. In this paper, we present a mechanism allowing packets to be routed through multiple paths in the network, helping to balance the traffic load while guaranteeing the in-order delivery. The proposed method combines the advantages of both deterministic and adaptive routing algorithms. The simple idea is to use different deterministic algorithms for independent flows. This approach neither requires reordering buffers nor limits packets to use a single path. The algorithm is simple and practical with negligible area overhead over dimension-order routing. The concept is investigated in both 2D and 3D mesh networks.

Keywords 2D mesh · 3D stacked mesh · In-order delivery

1 Introduction

Networks-on-chip (NoCs) is an alternative approach to the conventional bus architecture, offering scalability and flexibility to the Multiprocessor system-on-chip

M. Daneshtalab · M. Ebrahimi (✉) · S. Dytckov · J. Plosila
University of Turku, Turku, Finland
e-mail: masebr@utu.fi

M. Daneshtalab · M. Ebrahimi
KTH Royal Institute of Technology, Stockholm, Sweden

(MPSoC) design [1]. This flexibility allows NoCs to be efficiently employed in 3D designs [2].

An NoC consists of routers and links which are connected to each other in a structural manner. Cores are connected to the corresponding routers using the network interfaces. Cores can be classified as masters (e.g., processors) and slaves (e.g., memory modules). Master cores initiate transactions by generating requests (i.e., read and write) and one or more slaves receive and execute them. Dependent requests form a flow while independent requests belong to different flows. This implies that requests within a flow are differentiated from each other using a request ID while independent flows have different flow numbers.

All requests belonging to the same flow are required to reach the slave core in-order and the responses should also arrive to the master core in the same order. For example, if two requests are issued and delivered from a master core, the response of the first request should be received earlier than the second request. On the other hand, there is no ordering requirement between the requests of different flows.

The in-order delivery is guaranteed if packets always follow the same path for each pair of cores. However, due to the path diversity in NoCs, packets may use different routes facing different congestion. This results in disturbing the ordering of packets, and on the other hand limiting packets to use specific paths leads to performance loss in NoCs.

Routing algorithms can be classified as source and distributed routing. In source routing [3], the entire path is established at the source node prior to injecting a packet into the network, while in a distributed routing the path is determined at each intermediate router while a packet traverses across the network [4]. Unlike distributed routing, in the source routing approach, the path information is carried by packets, thus routers do not make routing computations. This characteristic results in a simpler routing unit and a faster communication. However, since in source routing packets carry the path information that is usually large, bandwidth and scalability become major challenges.

Distributed routing can be classified into deterministic and adaptive approaches [5]. In deterministic routing, a fixed path is used for each pair of nodes. Although deterministic routing is a predominant approach due to its simplicity, it degrades performance as the traffic load cannot be efficiently distributed. The simplest deterministic routing method is dimension-order routing. The dimension-order routing algorithm routes packets by crossing dimensions in a strictly increasing order, reducing to zero the offset in one dimension before routing in the next dimension. Deterministic routing algorithms perform well with uniform traffic patterns while they are very inefficient under non-uniform traffic. It is commonly believed that using deterministic routing, the in-order delivery is guaranteed. However, it is not a correct interpretation when virtual channels are used. This is due to the fact that packets from the same source may reach the destination at different times by facing different congestion in various virtual channels. In adaptive routing algorithms, a packet is not restricted to a single path when traversing from a source node to its destination [6]. So, adaptive routing algorithms can decrease the probability of routing packets through congested regions. Since packets use different routes, they may reach the destination in an arbitrary sequence. Thereby, the in-order delivery should be handled separately when exploiting adaptive routing. The basic assumption of these adaptive algorithms is that the reordering of packets

is handled at receivers. In addition, for preventing the overflow of reordering buffers, a control mechanism is required to manage the number of outstanding packets in the network. In other words, the number of delivered packets is limited by the size of reordering buffers. Based on this assumption, a simple analysis in [7] shows that by allocating the buffer space for only ten packets at the end nodes, the area and power consumption is increased by around 59 and 43 %, respectively, in a 4×3 mesh network. In reality, reorder buffers should keep more than 50 packets in an 8×8 network, imposing a large area and power consumption.

Therefore, either the size of reordering buffers should be enlarged enough to accommodate all packets or the number of transmitted packets from each node should be restricted. The former imposes a large area overhead whilst the latter underutilizes the network resources.

To overcome the out-of-order issue, all packets can be delivered to the destinations by following the same paths and using the same virtual channels. Although this is an efficient method under uniform traffic, the load balancing becomes a major issue when traffic is non-uniform. Considering the fact that real traffic patterns have mostly a non-uniform nature, a path-diverse in-order delivery approach is demanded for a better efficiency and performance.

In short, adaptive routing algorithms are efficient when out-of-order delivery is acceptable by the application or providing a large reordering buffer and a control mechanism at each node is not a challenging problem in terms of complexity, cost, and area. However, these requirements may not be provided by every system due to the cost and area limitations.

In this paper, we present an in-order delivery approach to both satisfy performance and guarantee the in-order delivery of packets without using reordering buffers. Unlike conventional methods in which packets are forced to use a single path, in the proposed approach, packets can be distributed over the network using different routes. The proposed In-Order Delivery (IDA) approach is discussed in both 2D (IDA-2D) and 3D (IDA-3D) networks. To the best of our knowledge, for the first time, we present a method to support the in-order delivery in 3D networks.

The paper is organized as follows. Section 2 presents the background and related work on the in-order and out-of-order delivery approaches. Fully adaptive routing algorithms are introduced in Sect. 3 for both 2D and 3D network, forming the base of the proposed in-order delivery approaches. In Sect. 4, the idea is introduced and the in-order delivery approach is explained. Results are reported in Sect. 5 while the conclusion is given in the last section.

2 Background and related work

Adaptivity in its different forms offers better performance and results in better optimization in many different areas [8,9]. In the field of NoCs, adaptivity can be discussed in routing algorithms where different methods try to suggest a better performance by bringing more adaptivity into the routing algorithms. Some examples in 2D and 3D networks are DyXY [10], FRA [11], MAR [12], and FAR [16]. Among them, DyXY and FAR are fully adaptive routing algorithms while FRA and MAR are partially adap-

tive methods. The routing selection of these algorithms is usually performed using the congestion status of the network. Many of them consider local traffic condition in the routing decision in which a router analyzes the congestion conditions of its own and adjacent routers to choose an output channel. Since this group of algorithms is not aware of the traffic condition beyond neighboring nodes, they might deliver packets through the regions that are highly congested. Routing decisions based on local congestion information may lead to an unbalanced distribution of traffic. Some other algorithms take more global information into account, reducing the probability of making a wrong decision. Besides the imposed area overhead, collecting global information is not an easy task. In addition, dynamic changes in traffic flow make it challenging to keep the global information updated. In general, congestion-aware routing algorithms improve the performance considerably compared with deterministic methods, but at the cost of a more complex routing logic and the need for congestion detection and propagation mechanism.

Although for many applications, the in-order delivery of packets is needed, this issue has rarely been investigated in NoCs [13]. EDVCA [14] deals with the problem of out-of-order delivery when the dimension-order routing (e.g., XY) is used. The solution of EDVCA is to dynamically allocate a virtual channel to a flow using a routing table at each router. This approach is extended in PDIOR [15] to improve the performance. PDIOR takes advantage of two routing algorithms (i.e., XY and YX) and two virtual channels; one virtual channel is allocated to the XY routing algorithm while the other one is utilized for the YX routing algorithm. In this method, a single flow of packets is divided into several flows. Each flow can be delivered using either the XY or YX algorithm. However, the flows cannot be simultaneously routed in the network. The sender should receive the acknowledgment of the last delivered packet of one flow before proceeding to the next one. Upon receiving the acknowledgment, packets of the next flow is delivered using either XY or YX and so on. When a flow contains many packets, it is efficient to break the flow into several flows; otherwise it may not be worth to split packets as a considerable amount of time is spent to receive acknowledgements. In addition, in the best case, packets are delivered through two paths (i.e., following XY or YX). However, path diversity plays an important role to achieve a better performance in this algorithm. In both EDVCA and PDIOR methods, a lookup table is required at each router to maintain the virtual channel number of different flows. Another in-order delivery method is proposed in [7] where packets are distributed on the partially non-intersecting paths and then the ordering is reconstructed in convergent nodes. This algorithm has several shortcomings as: 1—it needs a lookup table to track packets of different flows. Thereby, large amount of information should be stored at routers. For an instance, per each source and destination, the information of several flows must be stored. 2—it imposes complexity and overhead due to finding non-intersecting paths for each pair of source and destination nodes; 3—out-of-order packets at convergent nodes cannot proceed until all prior packets arrive, resulting in blockage of packets. 4—the maximum number of non-intersecting paths (i.e., the path diversity) in 2D NoCs is two paths, limiting the performance.

Source routing is another solution for the in-order delivery issue. In source routing, packets carry the whole path information and no routing decision is made at

intermediate routers. The main drawbacks of source routing approaches are the limited performance, channel bandwidth requirement, and scalability. The reason is that source routing approaches are deterministic and they require a large packet header to carry the path information. In source routing, the number of routing bits is proportional to the diameter of the network. The presented algorithm in [3] gets a profile of the traffic off-line and based on that, it finds efficient paths for communicating among cores. This approach is efficient for application-specific NoCs rather than the general-purpose NoCs.

3 Fully adaptive routing algorithms in 2D and 3D NoCs

The routing algorithms proposed in this paper are based on the rules of fully adaptive routing algorithms. In fully adaptive routing algorithms, all minimal paths can be taken between a pair of source and destination nodes. There are several fully adaptive routing algorithms in 2D and 3D mesh networks, among them DyXY and FAR offer the minimum number of virtual channels which we select in this manuscript.

3.1 DyXY Routing algorithm in 2D NoCs

In the dynamic XY (DyXY) routing algorithm [10], at each router, X or Y dimension can be selected without any limitation as long as the selected X and Y are in the shortest paths. However, this freedom to select between X and Y dimensions may result in deadlock and thereby DyXY needs a mechanism to guarantee deadlock freedom. For this purpose, DyXY utilizes one and two virtual channels along the X and Y dimensions, respectively. In this method, the network is partitioned into increasing and decreasing subnetworks as shown in Fig. 1a. The increasing subnetwork covers the positive direction of X and the first virtual channel in the Y direction while the decreasing subnetwork contains the rest of the channels (The negative direction of X and the second virtual channel in the Y direction). Therefore, if the destination router is to the east of the source, the packet will be routed through the increasing subnetwork. Similarly, if the destination router is to the left of the source, the packet will be routed through the decreasing subnetwork. In the case that the source and destination are in the same column, the packet can be routed using either subnetwork.

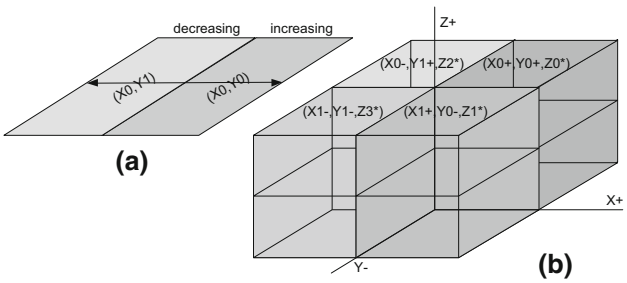

Fig. 1 Virtual channel allocation in (**a**) 2D (**b**) 3D network

3.2 FAR routing algorithm in 3D NoCs

FAR [16] is a fully adaptive routing algorithm in the 3D networks. FAR employs two, two, and four virtual channels along the X, Y, and Z dimensions which is the minimum number of virtual channels to provide fully adaptiveness in 3D networks. In the FAR algorithm, the network is partitioned into four subnetworks (Fig. 1b) as: $((X^+)(Y^+)(Z^*), (X^+)(Y^-)(Z^*), (X^-)(Y^+)(Z^*)$, and $(X^-)(Y^-)(Z^*))$ where "+", "−" represent the channels along the positive and negative directions, respectively, and "*" stands for both positive and negative directions. The deadlock freedom can be proved by dividing the network into virtual networks and guaranteeing that each of them uses a separate set of virtual channels. The virtual channel assignment of FAR is as follows: $((X0^+)(Y0^+)(Z0^*)$, $(X1^+)(Y0^-)(Z1^*)$, $(X0^-)(Y1^+)(Z2^*)$, and $(X1^-)(Y1^-)(Z3^*))$ where the numbers indicate the virtual channel numbers.

4 IDA: an in-order delivery approach

4.1 The need for in-order delivery

As it is already mentioned, the in-order delivery means that packets belonging to the same flow reach the destination in an in-order fashion. For example, in Fig. 2, there are three independent flows from the source node 0 to destination 7, 14, and 9. Each flow contains various numbers of packets in which they can be delivered to the destination continuously or at different times. In this example, the flow1 (F1), flow2 (F2), and flow3 (F3) are originated at the source node 0. Nine, four, and seven packets are delivered for destination 7, 14 and 9, respectively. All packets within a flow are dependent and should reach the destination in-order while packets of different flows are independent of each other. In this example, the dimension-order routing algorithm, XY, is employed to deliver packets. Following the XY routing algorithm, at the source node 0, packets of all three flows are sent through the X dimension (i.e., the east output port). Similarly, at the node 1, packets of flow1 and flow2 request the X dimension again. This is obvious that employing the XY algorithm creates congestion in the X dimension (e.g., router 1 and 2, and the corresponding links) while the other possible paths are underutilized (e.g., no packet has been routed through the node 4).

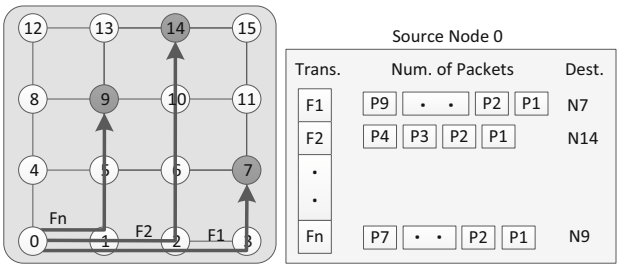

Fig. 2 In-order delivery in the 2D mesh network

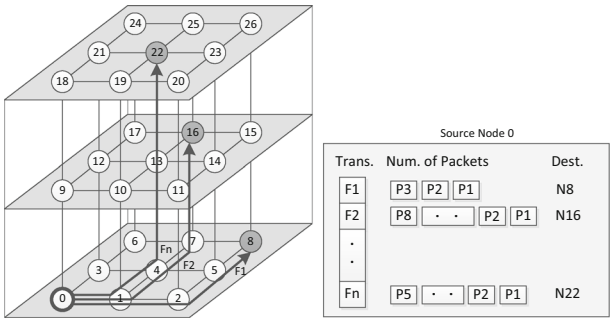

Fig. 3 In-order delivery in the 3D mesh network

As shown in Fig. 3, similar issues arise in the 3D network when using the XYZ routing algorithm. Therefore, traditional methods of supporting the in-order delivery suffer from low performance due to inability to distribute traffic over the network.

4.2 In-order delivery using multiple deterministic routing algorithms in 2D and 3D NoCs

The proposed method, named in-order delivery approach (IDA), diminishes the disadvantages of using deterministic methods to support the in-order delivery. Since different flows are independent of each other, they can use different routing algorithms. So, the idea behind IDA is to simply assign a deterministic routing algorithm to each flow instead of one algorithm to all flows. Some characteristics of IDA are as: 1—multiple routing algorithms are able to work together without forming any cycle in the network; 2—no additional virtual channel is needed to integrate different algorithms; 3—the general algorithm of combining different algorithms is simple, cost and power efficient.

Now, we need to find several routing algorithms to be assigned to different flows keeping in mind that routing algorithms should be compatible with each other without forming deadlock. We first start with IDA in the 2D network (IDA-2D) and then we investigate IDA in the 3D network (IDA-3D). For IDA-2D, we take advantage of a fully adaptive routing algorithm, DyXY. This algorithm utilizes one and two virtual channels along the *X* and *Y* dimensions, respectively. Since DyXY is a fully adaptive routing algorithm, packets can be routed through all available minimal paths. For example, in one path, a packet may switch between *X* and *Y* dimensions consecutively (repetitive-XY or repetitive-YX pattern), or it may be routed in the X dimension until the distance reaches zero and then the Y dimension (i.e., XY pattern). Depending on the distance between the source and destination routers, different patterns might be taken by packets such as XXY or YYYXYXX. Let us consider an example in Fig. 2 where a packet is sent from the source 0 to the destination 14. Using DyXY routing, packets may take different minimal routes as path: {0,1,5,6,10,14} having the pattern XYXYY or path: {0,4,8,9,13,14} with the pattern YYXYX. These examples show that normally by respecting the virtual channel assignment rules of DyXY (explained in Sect. 3.1), different patterns are taken in the network without forming a cycle.

Unlike DyXY where different patterns are dynamically taken by packets, in IDA different patterns are dynamically assigned to flows but packets of a flow follow the same pattern at a time. Thereby, one flow can follow the YYXYX routing algorithm while another flow may take the XYXYY routing algorithm. In this approach, a flow is assigned to a routing algorithm and the path information is stored in the header flit of the packet.

There are two main challenges with this approach. First, for a large network size, the path information may not fit in one header flit. Second, there is an overhead in finding a path among all the possible minimal routes and assigning it to a flow. Our solution is to use a limited number of alternative paths between the source and destination nodes. In IDA-2D, four deterministic routing algorithms are selected to be used in the network simultaneously as XY, repetitive-XY, YX, and repetitive-YX. Considering the source node 0 and the destination node 14 in Fig. 2, a flow is assigned to the deterministic routing XY {0,1,2,6,10,14}, another flow is assigned to the routing algorithm repetitive-YX {0,4,5,9,10,14}, and so on. For this specific source and destination pair, using these four different routing algorithms, traffic is distributed over thirteen links out of seventeen links in the minimal path.

In the 3D network, to find different routing algorithms which are compatible with each other, similar to 2D network we take advantage of a fully adaptive routing algorithm, FAR. Using FAR, packets can be routed over all available minimal paths between every source and destination pair. For example in Fig. 3, when the source and destination are located at the node 0 and 16, respectively, the available paths are: {0,1,4,7,16}, {0,1,4,13,16}, {0,1,10,13,16}, {0,3,4,7,16}, {0,3,4,13,16}, {0,3,6,7,16}, {0,3,6,17,16}, {0,3,12,13,16}, {0,3,12,17,16}, {0,9,10, 13,16}, {0,9,12,13,16}, {0,9,12,17,16}. All of these paths can be safely taken by FAR as it is a fully adaptive routing algorithm. Now, each of these routes can be deterministically assigned to a flow but dynamically over time. For example, a flow can be assigned to the deterministic routing XYZ (similar to the path: {0,1,4,7,16}) while another flow can use the deterministic routing YXYZ (similar to the path: {0,3,4,7,16}). When a flow delivers its entire packets, it can be assigned a new routing algorithm. Obviously, the mentioned routing algorithms can work together in the network without creating any cycle as different packets in FAR can use these paths and the algorithm remains deadlock free. The information of a single path should be stored in the header flit and carried by all packets within that flow. In this way, all packets of the same flow use a single path, even if they are generated at different times.

In IDA-3D, six deterministic routing algorithms are selected to be used in the network simultaneously: XYZ, XZY, YXZ, YZX, ZXY, and ZYX. Note that, in all of the selected algorithms, the offset reaches zero in one dimension before routing in the next dimension. This selection provides scalability and simplifies the overall algorithm as will be discussed in Sect. 5.3. Using the same example as in Fig. 3, a flow is assigned to the deterministic routing XYZ (similar to the path: {0,1,4,7,16}), another flow is assigned to the routing algorithm ZYX (similar to the path: {0,9,12,17,16}), and so on. For this specific source and destination pair, using these six different routing algorithms, traffic is distributed over almost all the links located in the minimal paths (except links (3, 4) and (12,13)).

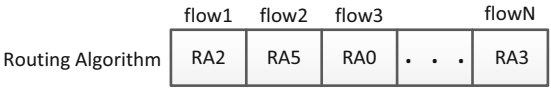

Fig. 4 Routing algorithms of different flows

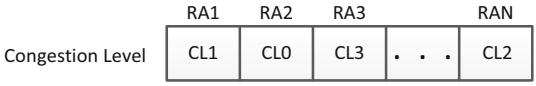

Fig. 5 Congestion level of the path for each routing algorithm

Packets of the same flow might be generated at different times but they must be delivered to the destination using the same routing algorithm assigned to that flow, guaranteeing in-order delivery. A small table, shown in Fig. 4, is required at network interfaces to maintain the routing algorithm assigned to each flow. For example, flow1 uses the routing algorithm 2 while flow2 follows the routing algorithm 5 to deliver packets. Upon receiving the last packet of a flow at the destination, an acknowledgement is sent back to the sender. This acknowledgement is responsible to collect the congestion information of the route traversed by the packet. A 2-bit field has been considered in the header flit for this purpose. This 2-bit value refers to four levels of congestion at a router, zero (00), low (01), medium (10), and high (11). At each intermediate router, the local congestion status is combined with non-local information stored in the header flit, and then the header flit is updated with the new congestion information. The computed congestion value of the router should be combined with those of carried by the packet. A 50–50 weight is assigned to the local and non-local congestion values. Finally, the newly computed value is stored in the header flit. When the acknowledgement packet arrives to the sender, the table of congestion values is updated based on the new congestion on the path (Fig. 5).

If no algorithm is assigned to a flow, the algorithm with the minimum congestion level is chosen. If there are multiple routing algorithms with the minimum value, one is selected by random. When all packets of a flow reach the destination and the acknowledgement is returned back, the corresponding field resets to zero and thus the flow is able to select another algorithm for the next period. An alternative approach is to statically assign a routing algorithm to each flow at design time, eliminating the usage of this table.

5 In-order delivery routing algorithm

5.1 Packet header format

To route packets efficiently, IDA employs different routing algorithms in 2D and 3D networks. In the 2D network four routing algorithms are chosen (XY, repetitive-XY, YX, and repetitive-YX) while in the 3D network, six algorithms are selected (XYZ, XZY, YXZ, YZX, ZXY, and ZYX). The reason for selecting these groups of routing algorithms for the 2D and 3D network is a better distribution of packets over the links.

The packet header format in the 2D mesh network is shown in Fig. 6. The packet type (T) indicates whether the algorithm is of type0 (i.e., XY or YX), or type1 (i.e.,

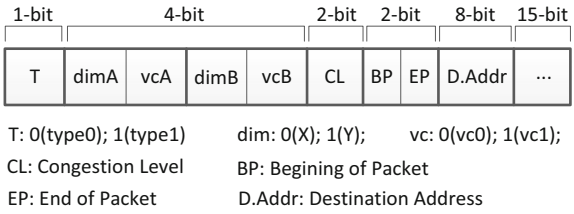

Fig. 6 Packet header format in the 2D network

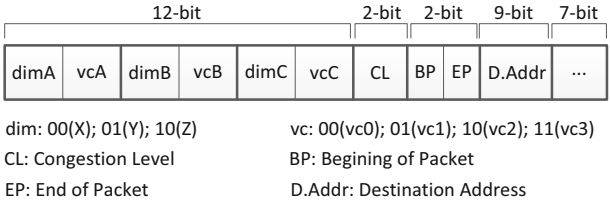

Fig. 7 Packet header format in the 3D network

repetitive-XY or repetitive-YX). Determining the packet type helps in simplifying the algorithm. DimA and dimB are mainly used to determine along which dimension the packet should be routed first. The allocation of virtual channels is done by respecting the DyXY rules. In the other words, eastward and westward packets use vc0 and vc1 along the Y dimension, respectively. There is no virtual channel used along the X dimension, so by default it is assigned to vc0. For example, if the packet is westward and the algorithm assigned to it is repetitive-YX, the fields are filled as: $T = 1$ (type1), $dimA = 1$ (Y dimension), $vcA = 1$ (vc1), $dimB = 0$ (X dimension), and $vcB = 0$ (vc0).

CL indicates the congestion level of the traversed path while BP and EP fields are used to determine the header and tail flits of the packet. In addition to this information, the destination address (D.Addr) is also stored in the header flit. Assuming the maximum network size of 16×16, 8-bit is required to encode every position in the network. In this form, the last 15 bits remain unused. IDA-2D employs four different algorithms. However, more number of algorithms can be employed in the network using 15 unused bits in the header flit and coding the algorithms.

In the 3D network, the six selected routing algorithms should be also encoded in the header flit. One way of encoding is to assign a number (i.e., 3-bit number between 000 and 101) to each algorithm. By implementing all six algorithms in the router at design time and extracting the header flit at run time, the corresponding routing algorithm is executed for each packet. This approach is scalable and simple, but the hardware cost is high as six different algorithms should be implemented in the routing unit. To achieve a low hardware overhead, we use a different approach which is designing a general algorithm in the routing unit to cover all six different routing algorithms. The header format of a packet is shown in Fig. 7. The first 12-bit of the header identifies the sequence of directions and virtual channels in which the message should take in the network. By assuming the maximum four virtual channels per dimension, the virtual channels can be encoded using two bits (i.e., vc0:00; vc1:01; vc2:10; vc3:11). Similarly, three dimensions are encoded in two bits (i.e., X:00; Y:01;

```
-- Current:X_C,Y_C; Destination:X_D,Y_D;
type<=Header[31];
dimA<=Header[30]; vcA<=Header[29];
dimB<=Header[28]; vcB<=Header[27];
CL<=Header[26 downto 25];
BP<=Header[24]; EP<=Header[23];
DAddr<=Header[22 downto 15];
Δx=X_D-X_C; Δy=Y_D-Y_C;
vcX<=vcA when dimA=X else vcB;
vcY<=vcA when dimA=Y else vcB;
-----
If (ΔX=0 and ΔY=0) select<=Local;
Elsif type=0 or inport=Local then
    If (dimA=X and ΔX!=0) then Select<=X,vcX;
    Elsif ΔY!=0 then Select<=Y,vcY;
    Else Select<=X,vcX;
ElsIf type=1 then
    If inport=east or inport=west then
        If ΔY!=0 then Select<=Y,vcY;
        Else Select<=X,vcX;
    Elsif inport=north or south then
        If ΔX!=0 then Select<=X,vcX;
        Else Select<=Y,vcY;
Else    An error has occurred.
End If;
```

Fig. 8 In-order delivery approach for the 2D network

Z:10). In addition to the encoded routing algorithms, CL, BP, EP, and the destination address (D.Addr) are also carried by the header. The last seven bits remain unused which can be reserved for the extension of the method to support more than six algorithms.

5.2 The IDA-2D routing algorithm

The in-order delivery algorithm for the 2D network (IDA-2D) is shown in Fig. 8. This algorithm covers the implementation of all four algorithms needed in IDA-2D. First, the header flit information is extracted. This information helps to detect the associated algorithm with the packet and its destination. The algorithms are from two types: type0 that determines dimension-order routing (XY or YX) while type1 refers to repetitive-XY or repetitive-YX. For the algorithms of type0, it is enough to know which dimension should be taken first. This information can be extracted from dimA. Thereby, if the algorithm is of type0, the distance along dimA should reach zero before proceeding to dimB. For the algorithms of type1, it is important to know from which dimension the packet arrives to the current router so that the packet is sent to the next dimension.

```
-- Current:X_C,Y_C,Z_C; Destination:X_D,Y_D,Z_D;
dimA<=Header[31 downto 30]; vcA<=[29 downto 28];
dimB<=Header[27 downto 26]; vcB<=[25 downto 24];
dimC<=Header[23 downto 22]; vcC<=[21 downto 20];
CL<=Header[19 downto 18];
BP<=Header[17]; BP<=Header[16];
DAddr<=Header[15 downto 7];
Δx=X_D-X_C; Δy=Y_D-Y_C; Δz=Z_D-Z_C;
-----
If (ΔX=0 and ΔY=0 and ΔZ=0) Select<=Local;
                -- Dimension A --
If dimA=X and ΔX!=0 then Select<=X,vcA;
Elsif dimA=Y and ΔY!=0 then Select<=Y,vcA;
Elsif dimA=Z and ΔZ!=0 then Select<=Z,vcA;
                -- Dimension B --
Elsif dimB=X and ΔX!=0 then Select<=X,vcB;
Elsif dimB=Y and ΔY!=0 then Select<=Y,vcB;
Elsif dimB=Z and ΔZ!=0 then Select<=Z,vcB;
                -- Dimension C --
Elsif dimC=X and ΔX!=0 then Select<=X,vcC;
Elsif dimC=Y and ΔY!=0 then Select<=Y,vcC;
Elsif dimC=Z and ΔZ!=0 then Select<=Z,vcC;
Else  An error has occurred.
End If;
```

Fig. 9 Assigning virtual channels to directions

5.3 The IDA-3D routing algorithm

The in-order delivery routing algorithm in the 3D network is shown in Fig. 9. The implementation of the routing unit is independent of the routing algorithms of different flows. In the other words, all routing algorithms are integrated into one single algorithm. By receiving a packet, the header flit information is extracted into dimA, dimB, dimC, vcA, vcB, vcC, CL, BP, EP, and D.Addr registers while the position of the current node (Current) is known by the node. The sequence of dimensions that a packet should traverse is dimA, dimB, dimC and their corresponding virtual channels are vcA, vcB, vcC. Since, the current router has no knowledge about the traversed path by the packet, it starts checking the header information from the left to the right side. First, it determines the dimension (e.g., X, Y, or Z) in which the parameter dimA is referred to. After specifying the dimension, the remaining distance along this dimension is checked (ΔX or ΔY or ΔZ). If the distance is not equal to zero, the packet is sent along this dimension using vcA; otherwise the next dimension (dimB) is examined for routing the packet and so on. As a result, the routing unit of IDA-3D not only is simpler than adaptive methods, but also faster due to eliminating extra routing effort.

In this paper, we put our focus on a mesh topology and using the minimum number of virtual channels. In addition, the number of concurrent algorithms is chosen to be four and six in 2D and 3D mesh network, respectively. However, the idea is general and

can be adjusted to different parameters. Thereby, more number of algorithms can be used and the idea can be applied to the networks with different dimensions, topologies, routing algorithms, and virtual channels.

6 Results and discussion

To evaluate the efficiency of the proposed in-order delivery approach (IDA), we developed a cycle-accurate NoC simulator based on wormhole switching for 2D and 3D network. The simulator calculates the average delay and power consumption for the message transmission. The simulator inputs include the array size, the routing algorithm, the link width, the buffer size, and the traffic type.

The arbitration scheme of the switch allocator is round-robin. The data width and the frequency were set to 32 bits and 1 GHz, respectively, and a buffer size of seven flits has been considered for each virtual channel. The packet size is randomly selected between three to eight flits. For the performance metric, we use the latency defined as the number of cycles between the initiation of a packet and the time when the tail flit reaches the destination.

IDA-2D is compared with a dimension-order routing algorithm (XY-Distributed Routing), fully adaptive routing algorithm (DyXY), a source routing algorithm (XY-Source Routing) and the in-order delivery routing algorithm (PDIOR) [15]. PDIOR uses two virtual channels along both the X and Y dimensions, meaning that it requires one more virtual channel than the proposed method IDA-2D. However, to have a fair comparison, we use one and two virtual channels for all algorithms. The PDIOR is adapted to this requirement by respecting the channel assignment rules explained in Sect. 3.1. The source routing approach employs the XY routing algorithm in which the information of the entire path is stored in the header flit. This information is used at intermediate nodes to reserve channels. The header flit contains the given output ports at intermediate routers. Since each router has seven output ports (i.e., L, E, W, N1, N2, S1, and S2), three bits are used to code the output channels of a router. PDIOR is an in-order delivery approach originally presented in the 2D network. In this algorithm, a flow can be delivered either using XY or YX algorithm.

IDA-3D is compared with dimension-order routing implemented in a distributed and source routing manner (XYZ-distributed routing and XYZ-source routing), and the fully adaptive routing algorithm (FAR). There is no other proposed in-order delivery routing algorithm in 3D networks to compare IDA-3D with. We use two, two, and four virtual channels along the X, Y, and Z dimensions, respectively, in all four methods. Using this assumption, the source routing algorithm requires four bits per router to determine the proper output channel.

The on-chip network, considered for experiment is formed by a typical router architecture including input buffers, a routing unit, a switch allocator and a crossbar. Each router has seven input/output ports, a natural extension from a 5-port 2D router by adding two ports to make connections to the upper and lower layers. In the 3D network, to estimate the power consumption as well as the power and delay values of vertical links, we used Orion [17]. The experiments are performed on 8×8 and $4 \times 4 \times 4$ mesh networks.

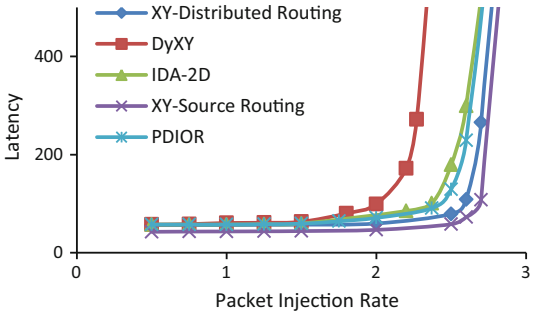

Fig. 10 Performance under uniform traffic profile in the 2D network

6.1 Performance analysis under uniform traffic profile

In the uniform traffic profile, each processing element (PE) generates data packets and sends them to another PE using a uniform distribution. In Fig. 10, the average communication latency as a function of the average packet injection rate is plotted for XY-Source Routing, XY-Distributed Routing, DyXY, PDIOR, and IDA-2D.

Under the uniform traffic profile, dimension-order routings make it possible to evenly distribute traffic over the network. It implies that they lead to the lowest latency. As can be seen in Fig. 10, XY-Source Routing approach has a lower latency than the distributed approach. This is due to the fact that in the source routing approach, the path is pre-computed and thus no routing decision is made at intermediate routers. However, by considering 3-bit information for determining the output channel at each router in the path, 45 bits is required to code the longest path in the 8×8 network. In the other words, the size of each flit in the source routing should be enlarged. For this purpose, we considered 64 bits for each flit and increased the link and buffer width to 64 bits as well.

Both PDIOR and IDA-2D are path-diverse deterministic routing and to some extend they are compatible with uniform traffic while IDA-2D offers more path diversity than PDIOR. The DyXY routing algorithm performs the worst because it is an adaptive approach and not efficient under uniform traffic while it performs well under non-uniform traffic pattern.

In Fig. 11, the average communication latency of FAR, IDA-3D, XYZ in both distributed and source routing are plotted. As can be seen in this figure, DyXYZ cannot perform as good as the other methods. This is due to the fact that in DyXYZ, the congestion information is taken into account which results in a non-uniform distribution of packets. This non-uniformity disturbs the distribution of packets in the uniform traffic profile. By similar deduction, the XYZ-based algorithms perform the best. Between two different implementations of dimension-order routing, source routing results in a better performance. But it is at the cost of a larger header size. In the $4 \times 4 \times 4$ network, the longest path in the network involves ten routers. Since 4-bit is needed to determine the output port at each router, in total 40 bits are required at the header flit which cannot be fit in the considered 32-bit flit. So, we increase the flit width to 64 and accordingly, the buffer size and link width are also increased. IDA-3D outperforms the adaptive approach FAR because IDA-3D does not disturb the uniform distribution of packets as FAR does.

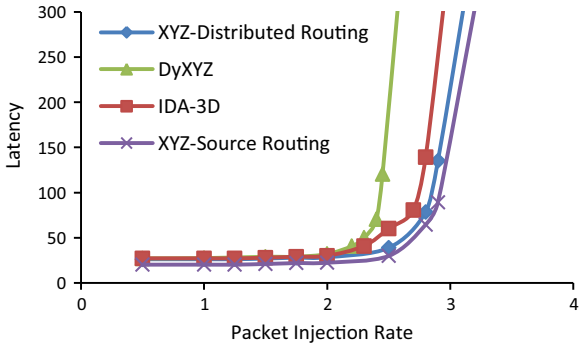

Fig. 11 Performance under uniform traffic profile in the 3D network

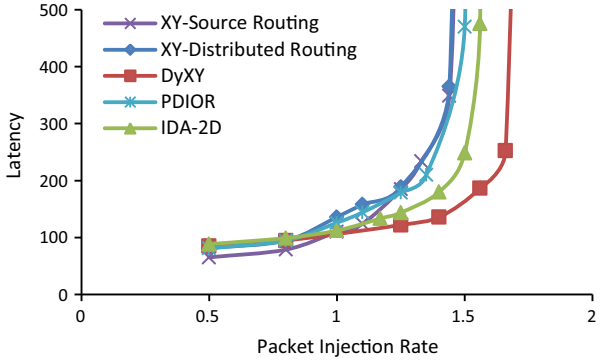

Fig. 12 Performance under hotspot traffic in the 2D network

6.2 Performance analysis under hotspot traffic profile

Under the hotspot traffic profile, some nodes are chosen as hotspots receiving an extra portion of traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of H, a newly generated message is directed to each hotspot node with an additional H percent probability. We simulate the hotspot traffic H = 10 % with a single hotspot router at (4, 4) in an 8 × 8 network and four hotspot nodes at positions (2, 1) and (3, 1) in layer 2 and the same positions in layer 3 in the 4 × 4 × 4 mesh network. For the 2D network, the performance of IDA-2D, XY-Distributed Routing, XY-Source Routing, DyXY, and PDIOR are plotted in Fig. 12. XY-Source Routing has the lowest delay in the low injection rate as no routing decision is made at intermediate routers. However, as it is already mentioned, 45 bits should be reserved in the header flit to encode the longest paths in the 8 × 8 network. Among all algorithms, DyXY leads to the best performance as the routing decision is made based on the congestion condition of the network. However, it comes at the cost of expensive reordering buffers at receivers. IDA-2D performs better than PDIOR and the dimension-order routing algorithms. In fact, the improvement is achieved by smoothly balancing the traffic over the network by routing packets through four different paths for each pair of source and destination nodes. This approach reduces the number of the

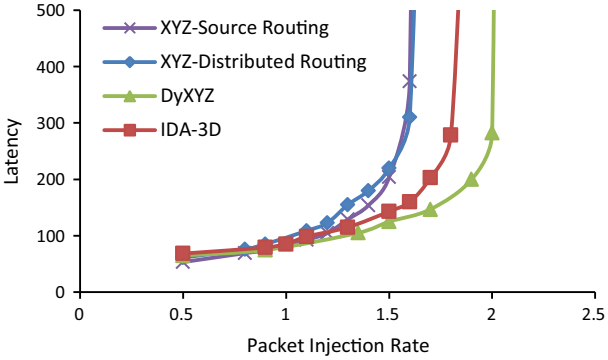

Fig. 13 Performance under hotspot traffic in the 3D network

hotspots and hence improving the performance. It is worth mentioning that DyXY can adaptively select among output ports based on the congestion condition in the input buffer of the neighboring routers. The performance of XYZ-Source Routing, XYZ-Distributed Routing, FAR, and IDA-3D are plotted in Fig. 13. The performance of IDA-3D is significantly better than those of the dimension-order routing algorithms. This is mainly because of the propagation of packets through different routes and distributing traffic over the network. In addition, IDA-3D takes the overall congestion of each path into account before assigning a routing algorithm to a flow. FAR dynamically makes the routing decision based on the congestion condition in the network which results in a better performance. However, FAR is not able to support the in-order delivery of packets. For such a support, a reordering buffer and a flow control mechanism are required.

The dimension-order routing algorithms are implemented in a distributed and a source routing way. XY- and XYZ-Source Routing deliver packets to their destinations in an in-order manner. However, XY- and XYZ-Distributed Routing are not able to guarantee the in-order delivery. Although all packets follow the same path, they can adaptively select between virtual channels based on the congestion condition. Thereby, packets might reach the destination in an out-of-order manner due to using different virtual channels. To address this problem, an approach similar to [14] is needed which assigns a virtual channel to a flow, requiring a lookup table at intermediate nodes.

6.3 Performance analysis under MPEG traffic profile in the 2D network

The XY-Source Routing, XY-Distributed Routing, DyXY, PDIOR, and the IDA-2D routing algorithms are evaluated under a MPEG4 decoder [18,19] mapped onto a 3×4 mesh topology. Figure 14 depicts the MPEG4 Decoder block diagram mapped onto 3×4 mesh topology. For each pair of source and destination nodes, 5–10 packets are randomly grouped into a single flow and are delivered using one of the algorithms in IDA-2D.

The performance gain of IDA-2D over XY-Source Routing, XY-Distributed Routing, and PDIOR is 16, 17, and 11 %, respectively (Fig. 15). This improvement is

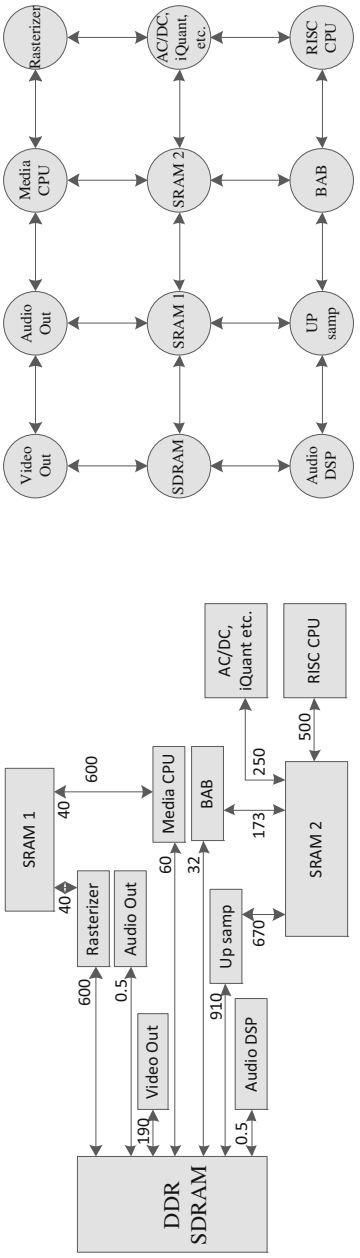

Fig. 14 **a** MPEG4 decoder block diagram, with communication BW annotated (in MB/s) and **b** its mapping onto a mesh topology [19]

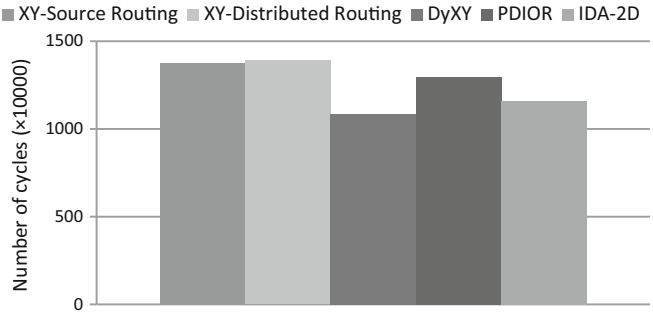

Fig. 15 Simulation results under a MPEG traffic profile

because of delivering packets over four different routes while in dimension-order routing and PDIOR, packets are limited to use one and two paths for each pair of source and destination nodes. Although IDA-2D takes the congestion information into account when choosing among the possible routing algorithms, DyXY dynamically uses the traffic information at intermediate routers. Thereby, it outperforms the IDA-2D by 6 %, but it is not able to support the in-order delivery of packets while requires large buffers at destinations to retrieve the ordering of packets.

6.4 Performance analysis under application traffic profile selected from SPLASH-2 in the 3D network

The GEMS full system simulator [20] is used as our simulation platform coupled with a cycle-accurate 3D NoC model. To know the real impact of the proposed method, we used traces from some application benchmark suites selected from SPLASH-2 [21]. Simulations are run on the Solaris 9 operating system based on the SPARC instruction. The adopted mapping strategy used in Solaris 9 is arbitrary mapping. Table 1 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 [22]. 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 64 nodes, 16 nodes are processors and the 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 [23].

The implemented memory hierarchy 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, 64KB, 2-way, 3-cycle access). The L2 cache is shared among all processors and split into banks (48 banks, 1MB each for a total of 48MB, 6-cycle bank access), connected via on-chip routers. The L1/L2 block size is 64B. The simulated memory hierarchy mimics SNUCA [24] while the off-chip memory is a 4GB DRAM with a 260-cycle access time. As an output, the simulator produces the communication latency for cache accesses. Figure 16 shows the average network latency of the real workload traces collected from the aforementioned system configurations, normalized

Table 1 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 three layers, unified, 48MB (48 banks, each 1MB), 64-bitline, 6-Clock
Cache coherence protocol	Token-based MOESI
Cache hierarchy	SNUCA
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Benchmarks	SPLASH-2
switch scheme	3D mesh with wormhole
Flit size	32 bits
SPLASH-2	Barnes, Cholesky, FFT, Ocean, Radix

to XYZ. As shown in this figure, under all application benchmarks, IDA-3D overcomes the XYZ routing algorithm while the implementation of IDA-3D is as simple as XYZ.

6.5 Area and power consumption

To assess the area overhead and power consumption of the proposed method, the whole platform including network interfaces and routers is synthesized using Synopsys Design Compiler with the 65nm LP CMOS technology from STMicroelectronics, while the backend design is performed with the Cadence Encounter tool. Depending on the technology and manufacturing process, the pitches of TSVs can range from 1 to $10\,\mu m^2$. In this work, the pad size for TSVs is assumed to be $5\,\mu m^2$ with the pitch of around $8\,\mu m^2$.

IDA-3D has the lowest area overhead as it does not require any reordering buffers at destinations. In IDA-3D, 23 bits are needed to store the path information in the header flit which is small enough to fit the flit size of 32 bits. The XYZ-Source Routing algorithm guarantees the in-order delivery of packets as the route and the virtual channel along the route are determined at the source node and all packets follow this route. However, as the required path information exceed the predetermined 32-bit flits, the buffer and link resources are doubled which results in the large area overhead.

Unlike XYZ-Source Routing, XYZ-Distributed Routing is not able to guarantee the in-order delivery as packets can dynamically switch between the virtual channels along the path. By considering the reordering buffers at network interfaces to accommodate 30 flits, a large area overhead will be imposed on this algorithm.

Similar to the XYZ-Distributed Routing algorithm, FAR cannot guarantee the in-order delivery while it requires reordering buffers.

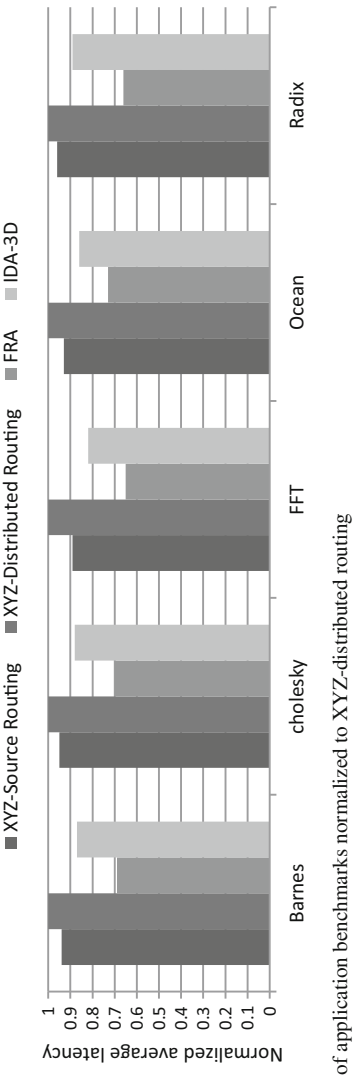

Fig. 16 Performance of application benchmarks normalized to XYZ-distributed routing

Table 2 Area overhead of different methods

Method	XYZ-Source Routing	XYZ-Distributed Routing	FAR	IDA
Area (mm^2)	27.04	18.21	29.62	18.56

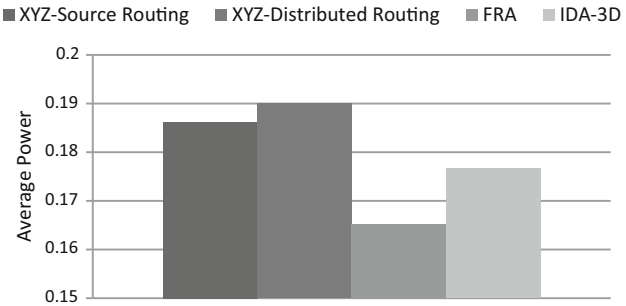

Fig. 17 Average Power Consumption of different methods

It is worth mentioning that for the performance evaluation of XYZ-Distributed Routing and FAR, unlimited buffer space has been considered in the network interfaces to obtain the best performance results for them. The area overhead of all methods has been listed in Table 2.

The power dissipations of all methods were calculated and compared under the hotspot traffic profile using the simulator based on the Orion [17]. The average power values are computed near the saturation point, 0.014. The power values of the network interfaces are integrated in the Orion functions. The typical clock of 1GHz is applied in the aforementioned network. The results for the average power under multicast traffic are shown in Fig. 17.

According to the results, XYZ-Distributed Routing has the highest power consumption which is 2, 7, and 13 % larger than that of the XYZ-Source Routing, IDA-3D, and the FAR scheme, respectively. This indicates that FAR distributes traffic better than the two other methods while IDA-3D performs better in balancing traffic than dimension-order routing. In fact, the better traffic distribution results in reducing hotspots and hence, lowering the average power.

7 Conclusion

In this paper, we proposed a new approach to guarantee the in-order delivery of packets while distributing packets over the network. In this method, the limitation of assigning a single routing algorithm to all flows is relaxed and thus each flow can be assigned to one routing algorithm. The idea has been applied to both 2D and 3D networks-on-chip. Using this method, several routing algorithms can work together in the network without blocking each other. The number of chosen algorithms in the 2D and 3D networks are four and six algorithms, respectively, and the implementations are optimized in each network. In addition, the routing logics of these algorithms are

as simple as a deterministic routing and they do not require any reordering buffers or a reordering mechanism to retrieve the ordering of packets. We can summarize that the in-order delivery is a basic requirement for many applications but it has rarely been investigated in the literature. We have reached a general in-order delivery approach in which the performance improvement is larger than the dimension-order routing while the simplicity, area and power consumption are nearly similar to it.

Acknowledgments This work was supported by VINNOVA (Swedish Agency for Innovation Systems) within the CUBRIC and ERoT projects and by Academy of Finland.

References

1. Palesi M, Daneshtalab M (2014) Routing algorithms in networks-on-chip. Springer, Berlin
2. Pavlidis VF, Friedman EG (2007) 3-D topologies for networks-on-chip. IEEE Trans Very Large Scale Integr VLSI Syst 15(10):1081–1090
3. Mubeen S, Kumar S (2010) Designing efficient source routing for mesh topology network on chip platforms. In: Proceedings of the 13th euromicro conference on digital system design: architectures, methods and tools, 2010, pp 181–188
4. Daneshtalab M, Ebrahimi M, Liljeberg P, Plosila J, Tenhunen H (2010) Input–output selection based router for networks-on-chip. In: IEEE computer society annual symposium on VLSI (ISVLSI), 2010, pp 92–97
5. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Tenhunen H (2011) Agent-based on-chip network using efficient selection method. In: Proceedings of 19th IFIP/IEEE international conference on very large scale integration (VLSI-SoC), 2011, pp 284–289
6. Wang X, Mak T, Yingtao Jiang MY, Daneshtalab M, Palesi M (2013) On self-tuning networks-on-chip for dynamic network-flow dominance adaptation. In: Proceedings of seventh IEEE/ACM international symposium on networks on chip (NoCS), 2013, pp 1–8
7. Murali S, Atienza D, Benini L, De Micheli G (2007) A method for routing packets across multiple paths in NoCs with in-order delivery and fault-tolerance gaurantees. VLSI Des 1–11. doi:10.1155/2007/37627
8. Ebrahimi M, Jahangireyan A (2013) Aerodynamic optimization of airfoils using adaptive parameterization and genetic algorithm. J Optim Theory Appl 162:257–271
9. Wang X, Yang M, Jiang Y, Mak T, Daneshtalab M, Palesi M (2014) On self-tuning networks-on-chip for dynamic network-flow dominance adaptation. ACM Trans Embed Comput Syst (TECS) 13(2):73–94
10. Li M, Zeng Q-A, Jone W-B (2006) DyXY—a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In: Proceedings of 43rd ACM/IEEE design automation conference, 2006, pp 849–852
11. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Tenhunen H (2011) Exploring partitioning methods for 3D networks-on-chip utilizing adaptive routing model. In: Proceedings of the fifth ACM/IEEE international symposium on networks-on-chip, 2011, pp 73–80
12. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Flich J, Tenhunen H (2012) Path-based partitioning methods for 3D networks-on-chip with minimal adaptive routing. IEEE Trans Comput 63:718–733
13. Palesi M, Holsmark R, Wang X, Kumar S, Yang M, Jiang Y, Catania V (2010) An efficient technique for in-order packet delivery with adaptive routing algorithms in networks on chip. In: Proceedings of the 13th euromicro conference on digital system design: architectures, methods and tools, 2010, pp 37–44
14. Devadas S, Cho MH, Shim KS, Lis M (2009) Guaranteed in-order packet delivery using exclusive dynamic virtual channel allocation, Technical Report, CSAIR-TR-2009-036, August 2009. Massachusetts Institute of Technology
15. Lis M, Cho MH, Shim KS, Devadas S (2010) Path-diverse in-order routing. In: Proceedings of international conference on green circuits and systems (ICGCS), 2010, pp 311–316
16. Ebrahimi M, Daneshtalab M, Plosila J (2013) In-order delivery approach for 3D NoCs. In: Proceedings of 17th CSI international symposium on computer architecture & digital systems (CADS), pp 93–98

17. Wang H-S, Zhu X, Peh L-S, Malik S (2002) Orion: a power-performance simulator for interconnection networks. In: Proceedings of 35th annual IEEE/ACM international symposium on microarchitecture (MICRO), 2002, pp 294–305
18. Ebrahimi M, Tenhunen H, Dehyadegari M (2013) Fuzzy-based adaptive routing algorithm for networks-on-chip. J Syst Archit 59(7):516–527
19. Tol EBVD, Jaspers EGT (2002) Mapping of MPEG-4 decoding on a flexible architecture platform. In: Proceedings of media processors, 2002, pp 1–13
20. Martin MMK, Sorin DJ, Beckmann BM, Marty MR, Xu M, Alameldeen AR, Moore KE, Hill MD, Wood DA (2005) Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Comput Arch News 33(4):92–99
21. Woo SC, Ohara M, Torrie E, Singh JP, Gupta A (1995) The SPLASH-2 programs: characterization and methodological considerations. In: Proceedings of 22nd annual international symposium on computer architecture, 1995, pp 24–36
22. Muralimanohar N, Balasubramonian R, Jouppi N (2007) Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In: Proceedings of the 40th annual IEEE/ACM international symposium on microarchitecture, 2007, pp 3–14
23. Park D, Eachempati S, Das R, Mishra AK, Xie Y, Vijaykrishnan N, Das CR (2008) MIRA: a multi-layered on-chip interconnect router architecture. In: Proceedings of international symposium on computer architecture, 2008, pp 251–261
24. Beckmann BM, Wood DA (2004) Managing wire delay in large chip-multiprocessor caches. In: Proceedings of the 37th annual IEEE/ACM international symposium on microarchitecture, 2004, pp 319–330