CATRA- Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks

Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku, Finland
{masebr,masdan,pakrli,juplos,hanten}@utu.fi

Abstract—**Congestion occurs frequently in Networks-on-Chip when the packets demands exceed the capacity of network resources. Congestion-aware routing algorithms can greatly improve the network performance by balancing the traffic load in adaptive routing. Commonly, these algorithms either rely on purely local congestion information or take into account the congestion conditions of several nodes even though their statuses might be out-dated for the source node, because of dynamically changing congestion conditions. In this paper, we propose a method to utilize both local and non-local network information to determine the optimal path to forward a packet. The non-local information is gathered from the nodes that not only are more likely to be chosen as intermediate nodes in the routing path but also provide up-to-date information to a given node. Moreover, to collect and deliver the non-local information, a distributed propagation system is presented.**

I. Introduction

System-on-chip (SoC) design is moving towards the integration of tens or hundreds intellectual property (IP) blocks on a single chip. As chip integration grows, the on-chip communication becomes performance bottleneck in high performance multiprocessor systems on chip (MPSoCs). The regular tile-based Network-on-Chip (NoC) architecture has been proposed as a solution to meet the performance and design productivity requirements of the complex on-chip communication infrastructure [1][2][3]. NoC provides an infrastructure for better modularity, scalability, fault tolerant and higher bandwidth compared to traditional infrastructures [1].

Routing algorithms are classified as deterministic and adaptive algorithms. The dimension-order routing algorithms route packets by crossing dimensions in strictly increasing (or decreasing) order, reducing to zero the offset in one dimension before routing in the next one. Adaptive routing has been used in interconnection networks to improve network performance and to tolerate link or router failure. In adaptive routing algorithms, a packet can travel from a source to a destination through multiple paths. Specifically, adaptive routing algorithms can be used to avoid congestion by adapting the routing decision to the network status.

Congestion may lead to increased transmission delay and power consumption, and thus limiting the performance of NoC. However, the performance can be improved by routing packets through less congested areas and flattening the distribution of traffic over the network. To alleviate congestion, some approaches consider local traffic conditions, i.e. the routing decision is made only based on congestion status of adjacent neighbors. However, these methods provide a limited view of the network condition; thereby they are not able to balance the traffic load across the network in a non-uniform or bursty traffic.

Globally adaptive routing algorithms mitigate this issue by considering network status beyond neighboring nodes. However, in the routing decision, they commonly incorporate the congestion information of the nodes in which a few percentage of packets pass through them or have out-dated information for the current node.

In this paper, we presented a routing method, named Congestion Aware Trapezoid-based Routing Algorithm (CATRA). This method takes advantages of local and non-local congestion information in the routing decision. CATRA method can increase the performance by efficiently balancing the network loads over the network. The contribution of the paper is threefold: the first idea is to involve the congestion status of the nodes that are more likely to be selected in the routing path. Based on this, we introduce an idea of monitoring the congestion status of the nodes in trapezoid positions. Moreover, when comparing the congestion value in two minimal directions, CATRA considers the congestion status of the nodes that not only locate in trapezoid positions, but also reside in the minimal quadrant defined by a source and destination pair. Therefore, CATRA can efficiently isolate congestion information of different applications. Furthermore, to allow a fair comparison between the congestion values in two minimal directions, we take into account the congestion value of the same number of nodes in each direction. The second idea of CATRA is to collect and distribute traffic information of distant nodes by utilizing distributed propagation system. This suggests relatively low wiring overhead, latency and power consumption required for propagating traffic information. Third idea is to utilize history-based congestion detection metric, avoiding abruptly change of the congestion flag from ‘1’ to ‘0’ or vice versa. The paper is organized as follows. Section II presents related works in congestion-oblivious and congestion-aware algorithms. The proposed scheme is discussed in Section III. The results are reported in Section IV while the conclusion is given in the last section.

II. Related Work

In congestion-oblivious algorithms, such as XY [4], West-First [5] and random [5], routing decisions are independent of the congestion condition of the network. This policy may disrupt the load balance since the network status is not considered, while the congestion may dramatically increase the service time of packets. Unlike congestion-oblivious methods, in congestion-aware algorithms, such as DyXY [6], BARP [7], Odd-Even [8], GOAL [9] and GAL [10], the selection is usually performed using the congestion status of the network [11]. Most of the congestion-aware algorithms consider local traffic condition; each router analyses the congestion conditions of its own and adjacent routers to choose an output channel. Routing decisions based on local congestion information may lead to an unbalanced distribution of traffic load. EDXY method is introduced in [12] where the congestion information of a router is propagated to its row and column nodes via separate wires. Non-local information provided by these wires is used only when the packet is one row or column away from the destination. In NoP [13] the routing decision is performed based on the congestion information of the nodes within two hops of the current node that are located in the minimal path to the destination. A well-known method named Regional Congestion Awareness (RCA) is proposed in [14] to utilize non-local congestion information in routing decision. In the RCA method, in order to provide global congestion information, the locally computed congestion value of a router is combined with those global signals propagated from downstream routers and the newly-aggregated value is transmitted to the upstream routers and so on. The shortcomings of NoP and RCA methods has been extensively discussed in [15] when presenting the DBAR method. Among all presented algorithms, DBAR suggests better performance since it utilizes both local and non-local congestion information in the routing decision while offering dynamic isolation among regions. The general aspects of DBAR are discussed in detail in Section III-E.

III. CONGESTION AWARE TRAPEZOID-BASED ROUTING ALGORITHM (CATRA)

A. *Passing-Probability of Packets through the Nodes*

For a given source and destination pair in an adaptive routing, some intermediate nodes forward more packets than the others. In general, central nodes forward significantly larger number of packets than edge nodes, since central nodes are accessible via more paths. One of the aims of CATRA method is to make the routing decision based on the congestion conditions of the nodes through which more packets are forwarded. For an example, consider a case in Fig. 1 where a packet is sent from the node 0 to the destination 35 while the network is not congested. At the source node, the packet can be sent through nodes 1 and 6 with 50% probability each. When packet arrives at node 1, there are two choices for the next hop, nodes 2 and 7. The probability that a packet passes either node is 25%. The arrived packet at both nodes 2 and 7 can be delivered to the node 8, thereby with a probability of 25% a packet transfers through node 7 to the destination, and so on. Similar to traditional techniques [14][15], the routing decision can be assisted by the status of the nodes explicitly in the row or column (i.e. nodes 1,2,3,4, and 5 in the row; and nodes 6,12,18,24, and 30 in the column); while probability of passing the packets through the nodes along a row or column are 50%, 25%, 12.5%, 6.25%, and 3.125%, respectively. However, by continuing along a direction, not only the passing probability of packets through the nodes approaches zero but also their congestion status would be out-dated for the node 0. In general, the traditional methods do not efficiently improve the load balance due to the lack of knowledge about the congestion status of the nodes to be likely passed in the routing path. In CATRA method, instead, the congestion status of the nodes in trapezoid positions is analyzed. Trapezoid positions for the X and Y directions are shown in Fig. 1. Packets delivered at node 0 may pass the nodes in X-trapezoid positions (i.e. nodes 1,2,8,3, and 9) with a probability of 50%, 25%, 25%, 12.5%, and 18.75%, respectively. So, the probability of the nodes 8 and 9 being passed by the packets from the node 0 is higher than either nodes 3 and 4 in the axes. Although the nodes 7 and 14 can be used in the routing path with a probability of 25% and 18.75%, respectively, their congestion status cannot affect the comparison result at node 0 as they have similar effect on congestion values in both directions. The other nodes are either far from the source node or have a little chance to be an intermediate node through which packets are forwarded.

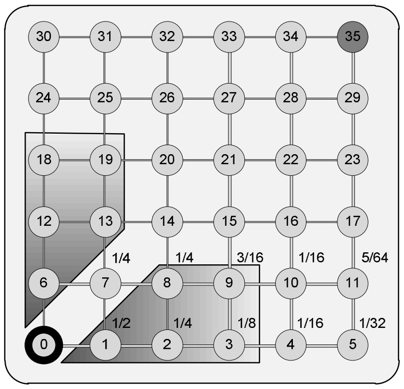


Fig. 1. Passing-Probability of Packets through Intermediate Nodes.

B. *Congestion Propagation Strategy*

To distribute the congestion information of the nodes reside in trapezoid positions, every four routers are connected to an agent (Fig. 2(a)). Each agent has two main tasks: 1- Collecting the congestion information from the attached routers (local routers) and distributing the information to the neighboring agents. 2- Receiving non-local information from neighboring agents and delivering it to local routers.

To explain the basic idea of the propagation strategy, let us consider an example in Fig. 2(b) where the congestion status of the nodes in trapezoid positions (i.e. nodes B, C, G, D, and H) are delivered to the node A. Congestion status of nodes H and D is sent to the agent 2. In the agent 2, the congestion status of nodes H and D should be aggregated with the incoming value from the downstream agent. However, since the agent 2 is at the rightmost column, it receives no information from the east direction. Therefore, the agent 2 combines the 2-bit information of nodes H and D with 1-bit with the value of zero, and delivered the value to the agent 1. At the agent 1, the received 3-bit congestion information is directly sent to nodes F and B. Meanwhile, the two most significant bits of the arrived value are OR-ed together and combined with the status of nodes G and C, and then the 3-bit information is delivered to the agent 0. Finally, arrived information at the agent 0 is sent to nodes A and E. Therefore, node A is informed about the congestion status of nodes C, G, D, and H, while the 1-bit information of node B is sent directly to the node A.

Fig. 2(c) illustrates a central agent which transfers the information from/to the four neighboring agents. As shown in this figure, the congestion value of nodes A and B are ORed together in the central agent and combined with the status of nodes D and E, and finally sent to the north agent. Similarly, the ORed value of nodes J and F(nodes G and C, nodes K and L) are merged with those of nodes I and E (nodes H and D, nodes H and I) before transmitting to nodes G and C (nodes J and F, nodes A and B). As shown in Fig. 2(d), an agent receives 3-bit congestion information from each neighboring agent. The incoming information from the east neighboring agent is sent to the westward nodes (i.e. nodes H and D). Similarly, the received information from the north, west, and south neighboring agents are delivered to southward, eastward, and northward nodes. Thus, at most two 3-bit wires should be connected from each agent to the surrounding nodes.

C. *Congestion Metrics*

All nodes receive 1-bit congestion information from each neighboring router and two 3-bit congestion statuses from each connecting agent. We describe the congestion metrics for computing the congestion conditions of neighboring and distant nodes as follow:

1) *Congestion computing of neighboring nodes*

Congestion-aware algorithms can take advantages of different metrics such as the number of free input buffers [6][7][12][13], available virtual channels [15], crossbar demand [14], or combinations of these factors [14]. In this paper, to estimate the congestion of neighboring routers, we have chosen the buffer availability at the corresponding input port in the next hop as congestion metric. However, since 1-bit wire is used to exchange information between neighboring nodes, the number of free buffer slots should be condensed into 1-bit. One solution is to activate the congestion bit whenever more than 75% of the buffer space is occupied. However, this introduces an abrupt change in the congestion value whenever the number of occupied buffer slots is increased from 4 to 5 (or decreased from 5 to 4 for a total of 8 slots). To diminish this problem, we employ a history based scheme capturing the threshold signals. Fig. 3 illustrates the congestion detection circuit where a 3-bit shift register is adopted to store the threshold signal 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 buffer is larger (smaller) than a threshold value, the threshold signal is assigned to one (zero) and stored in the shift register. According to this strategy, the congestion flag is raised if and only if two or three bits of the shift register are 1.

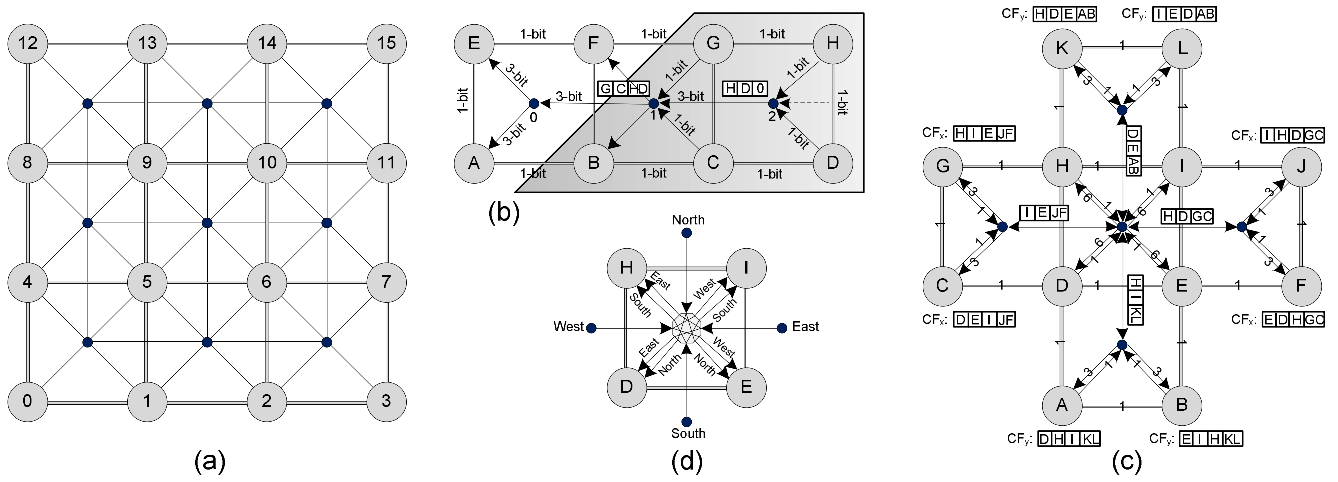


Fig. 2. (a) Congestion propagation network (b) Congestion propagation of nodes in trapezoid positions (c) transferring information among agents (d) delivering information from agents to local routers.

2) *Congestion computing of distant nodes*

To measure the congestion condition of distant nodes, the total number of available buffer slots at each router is used as the congestion metric, so that if the total number of occupied slots in a router is larger than a threshold value, the node is regarded as congested. To compute the optimal threshold value, we performed an empirical evaluation on a wide range of buffer sizes and traffic loads. Based on our experimental results, the best results were obtained when the threshold value was set to 60%. We use a similar history-based strategy as illustrated in Fig. 3 to ignore the transient changes in the congestion condition.

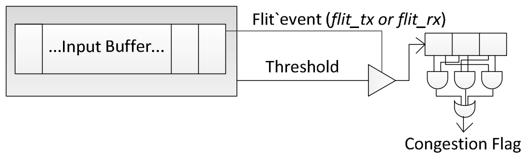
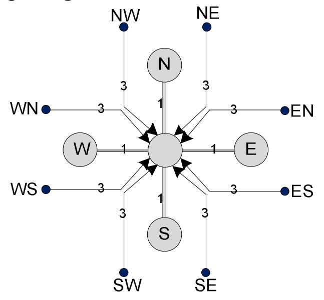


Fig. 3. Congestion detection circuit

D. *Congestion Computing Algorithm (CCA)*

By distributing congestion information over the network, routing decision can be assisted by the local and non-local congestion information received from the nodes in trapezoid positions. As indicated in Fig. 4, a node receives 1-bit congestion information from each neighboring router and two 3-bit congestion statuses from each connecting agent. For an instance, if the destination is in the northeast of the source node, 1-bit congestion status of the north neighbor is concatenated with 3-bit congestion value of the northeast agent (NE) and the result is stored in a 4-bit register named CR_y. In a similar manner, the value of CR_x is obtained. Note that, congestion value of the neighboring routers is used as the most significant bit of CR_y or CR_x. In general, the congestion value of the nodes in trapezoid positions are assigned to the bits of CR_x and CR_y registers based on the passing probability of the packets through the nodes and their distances to the source node. As illustrated in Fig. 2(c), each bit in CR_x and CR_y signifies congestion status of a specific node in a trapezoid position. This suggests that for a given source and destination pair, the congestion status of nearby routers with a high packet passing probability influences routing decision more than long-distance nodes. The pseudo code of CCA algorithm is presented in Fig. 5. According to CCA, when a packet is in the same row or column as the destination, the packet is sent to the appropriate direction (Lines 1-8). However, if the packet is one hop apart from the destination row or column, only the congestion conditions of the adjacent routers is used in the routing decision. In this situation, if the congestion status of the neighboring router in the X direction is similar to that of in the Y direction, the packet is sent to the neighboring node that probably is not located in the destination row or column. This selection retains adaptively for the packet in the next hop (Lines 9-22). When the packet is two hops away from the destination row or column, the algorithm ignores the status of the nodes that are not located in the minimal path to the destination (Lines 23-28). In other cases, CR_x and CR_y are compared together (Lines 29-34). Since, in all cases, the congestion conditions of the nodes in the minimal paths are considered, this algorithm offers a well suited approach to application's isolation, when different applications are mapped to multiple regions of the network.



Destination Position	CR_y	CR_x
North-East	N & NE	E & EN
North-West	N & NW	W & WN
South-East	S & SE	E & ES
South-West	S & SW	W & WS

Fig. 4. Received congestion information by a router

Three examples of CCA method are presented in Fig. 6. In Fig. 6(a) a packet is sent from the source node 15 to the destination 2. At first, the routing function at the node 15 returns the west and south neighbors as the possible choices to forward the packet. Since the packet is one hop away from the destination column, only the congestion value of the adjacent nodes 14 and 11 are compared together. The reason for this policy is to ignore the congestion value of the nodes that are not located in the minimal path (i.e. ignoring the information of nodes 13,9,12, and 3) and to compare the congestion information of the same number of nodes in each direction (i.e. ignoring the information of nodes 7,6,3, and 2). If the flag of both nodes 14 and 11 are one or zero, in CCA algorithm, the packet is delivered to the node 14, since otherwise the packet has to be routed to node 11, losing alternative choices in the remaining path. Fig. 6(b) shows another situation where a packet is sent from the source 15 to the destination 1. Since nodes 12 and 8 are located outside of the minimal path, their congestion values are ignored in the routing decision, and for a fair comparison, the congestion status of nodes 2 and 3 are also disregarded. Finally if the destination is located at node 0 (Fig. 6(c)), the congestion information of all nodes at the trapezoid positions is used.

E. *Comparing CATRA and DBAR Approaches*

DBAR method is described in [15] under the assumption of 8×8 mesh network. In this method, each router has two 9-bit registers, named congestion-X and congestion-Y. These registers are used to store the congestion information received from X and Y directions. Each entry in congestion-X (congestion-Y) is allocated to congestion status of a router in the row (column). At each router, the local congestion value is merged with the incoming congestion information from the downstream router in appropriate direction, and then the aggregated value is sent to the upstream router. DBAR method ignores the congestion value of the nodes that are located outside of the minimal path defined by a source and destination.

```
IF (Δx=0 or Δy=0) THEN
  IF (Δx=0 and Δy=0) THEN
     Select <= Local;
  ELSIF (Δx=0) THEN
     Select <= Y-dir;
  ELSE
     Select <= X-dir;
  END IF;
ELSIF (Δx=1 or Δy=1) THEN
  IF (Δx=1) THEN
     IF (CR_x(3)>=CR_y(3)) THEN
       Select <= Y-dir;
     ELSE
       Select <= X-dir;
     END IF;
  ELSIF (Δy=1) THEN
     IF (CR_y(3)>=CR_x(3)) THEN
       Select <= X-dir;
     ELSE
       Select <= Y-dir;
     END IF;
  END IF;
ELSIF (Δx=2 or Δy=2) THEN
   IF (CR_x(3 downto 1)>=CR_y(3 downto 1)) THEN
      Select <= Y-dir;
   ELSE
      Select <= X-dir;
   END IF;
ELSE
   IF (CR_x>=CR_y) THEN
      Select <= Y-dir;
   ELSE
      Select <= X-dir;
   END IF;
END IF;
```

Fig. 5. Congestion Computing Algorithm

1) *Congestion Information Utilization*

One drawback of DBAR method relates to an unfair comparison when a packet has different distances along X and Y directions to the destination node. For an instance in Fig. 6 (a), the problem of unfair decision can be observed when a packet is sent from the source 15 to destination 2. DBAR considers the congestion value of the nodes along the X and Y directions that reside in the minimal path to the destination node. However, if the congestion conditions of both nodes 11 and 14 are zero or one, the congestion status of nodes 7, and 3 in the Y direction are compared with the value of zero in the X direction, thereby the packet is probably sent to the node 14. Moreover, this unfair comparison leads to losing the routing adaptivity for the remaining path to the destination. CATRA deals with this problem by taking into consideration the congestion information of the same number of nodes for each direction.

2) *Passing- Probability of Packets through the Nodes*

Another shortcoming of DBAR method is the lack of knowledge about the status of the nodes which are not located along axes. As already mentioned, packets arriving from a given node may traverse through five nodes along a direction with a passing probability of 50%, 25%, 12.5%, 6.25%, and 3.12% while their distances to the source node are 1, 2, 3, 4, and 5 hops, respectively. Therefore, by moving along the axes, not only the passing probability approaches zero, but also due to the long distances from the source node, the router may make decision based on out-dated information. In contrast, CATRA involves the congestion conditions of the nodes with a passing probability of 50%, 25%, 25%, 12.5%, and 18.75% while located within 1, 2, 3, 3, and 4 hops from the source node.

3) *Wiring Overhead*

In $n\times n$ mesh network there are $2n(n\text{-}1)$ links with α average length (horizontal and orthogonal links). In DBAR method, the congestion wires have n-bit width in $n\times n$ mesh network. Therefore, the total wiring requirements of DBAR method can be calculated by:
$\text{Overhead}_{DBAR}=2(n^{\wedge}2)(n\text{-}1)\alpha$.

In the following, we list the wiring requirements of CATRA method:
$2(n)(n\text{-}1)$ links with 1-bit width and α length to propagate information among neighboring nodes in one direction.
$2(n\text{-}1)(n\text{-}2)$ links with 3-bit width and α length for distributing packets between agents in one direction.
$2(n\text{-}1)^{\wedge}2$ links with 1-bit width and $\sqrt{2}\alpha$ average length for transferring congestion status from local routers to agents (diagonal links).
$2(n\text{-}1)^{\wedge}2$ links with 3-bit width and $\sqrt{2}\alpha$ average length for transferring congestion status from agents to local routers (diagonal links).
Therefore, the total wiring overhead of CATRA method is:
$\text{Overhead}_{CATRA}= 4(n)(n\text{-}1)\alpha + 12(n\text{-}1)(n\text{-}2)\alpha + 2((n\text{-}1)^{\wedge}2)(\sqrt{2}\alpha) + 6((n\text{-}1)^{\wedge}2)(\sqrt{2}\alpha)$
On top of that, in $n\times n$ mesh network, given a channel width of 128 bits [14][15], the wiring requirements for transmitting data packets is:
$\text{Overhead}_{DataPackets}=2\times 128\times(n)(n\text{-}1)\alpha$.

According to these formulas, in 8×8, 16×16 and 25×25 mesh networks, the wiring overhead of DBAR method is 3.125%, 6.25%, and 9.76%, respectively, while in CATRA method the wiring overhead is 5.91%, 6.44%, and 6.63%, respectively. This suggests that the wiring overhead of DBAR method increases significantly as the network scales without any gain in performance. That is due to the fact that long wires are required to transfer the congestion information all over the rows and columns. This information may never be used by the nodes if the traffic is isolated by different applications in the network. On the contrary, CATRA propagates optimal amount of local and non-local congestion information by utilizing distributed approach without using global long wires.

IV. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed routing scheme, DBAR and NoP routing protocols are also implemented. For fairness, CATRA, DBAR and NoP utilize a fully adaptive routing function based on MAD-Y [16]. A NoC simulator is developed with VHDL to model all major components of the on-chip network and simulations are carried out to determine the latency characteristic of each network. For all the routers, the data width is set to 32 bits. The congestion threshold value is set to four meaning that the congestion condition is considered when four out of six buffer slots are occupied. As a performance metric, we use latency defined as the number of cycles between the initiation of a message transmission issued by a Processing Element (PE) and the time when the message is completely delivered to the destination PE. The request rate is defined as the ratio of the successful message injections into the network interface over the total number of injection attempts. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. Two synthetic traffic profiles including uniform random and hotspot, along with SPLASH-2 [17] application traces are used. For all routers, the data width and the frequency is set to 32 bits and 1GHz, respectively, which leads to a bandwidth of 32 Gb/s. Moreover, the packet length is uniformly distributed between 1 and 5 flits.

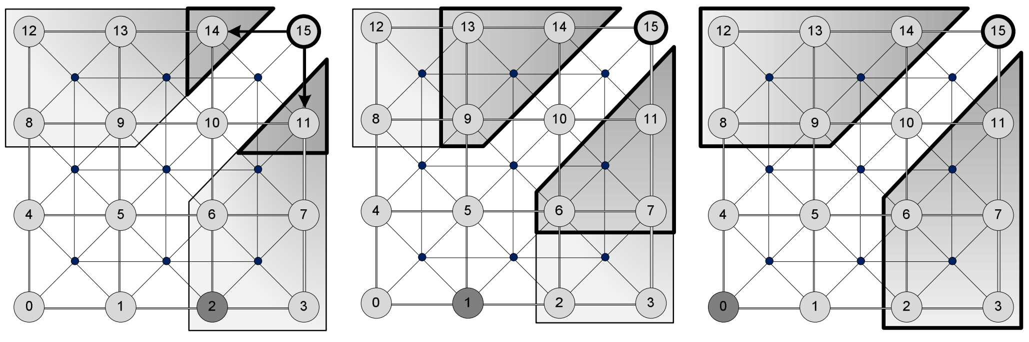


Fig. 6. Examples of CCA algorithm

A. *Performance Analysis*

1) *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 [5][8]. The mesh sizes are considered to be 8×8 and 14×14. In Fig. 7, the average communication delay as a function of the average packet injection rate is plotted for both mesh sizes. As observed from the results, CATRA leads to the lowest latency. This is due to the fact that CATRA can distribute traffic more efficiently than the other two schemes since the routing unit decision is based on the congestion status of the nodes through which more packets may pass toward corresponding destinations. Moreover, CATRA method compares the congestion information of the same number of nodes in each direction, while DBAR method suffers from an unfair comparison between the congestion values.

2) *Hotspot Traffic Profile*

Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the 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 with a single hotspot node at (4, 4) and (7, 7) in the 8×8 and 14×14 2D-meshes, respectively. The performance of each network with H = 10% is illustrated in Fig. 8. As observed from the figure, the proposed routing scheme achieves better performance compared to the other schemes.

Table 1. Hardware implementation details.

Network platforms	Area (mm^2)	Power (W) dynamic & static
NoP	6.970	2.75
DBAR	6.791	2.46
CATRA	6.843	2.51

3) *Application Traffic Profile*

Application traces are obtained from the GEMS simulator [18] using some application benchmark suites selected from SPLASH-2. We use a 64-node network configuration: 20 processors and 44 L2-cache memory modules. For the CPU, we assume a core similar to Sun Niagara and use SPARC ISA [19]. Each L2 cache core is 512KB, and thus, the total shared L2 cache is 22MB. 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 (44 banks, 512 KB each for a total of 22 MB, 6-cycle bank access), connected via on-chip routers. The L1/L2 block size is 64 B. Our coherence model includes a MESI-based protocol with distributed directories, with each L2 bank maintaining its own local directory. The simulated memory hierarchy mimics SNUCA 0 while the off-chip memory is a 4 GB DRAM with a 220-cycle access time. Fig. 9 shows the average packet latency across four benchmark traces, normalized to NoP. CATRA provides lower latency than other schemes and it shows the greatest performance gain on Radix with 26% reduction in latency. The average performance gain of CATRA is up to 20% across all benchmarks vs. NoP and 13% vs. DBAR.

B. *Hardware Analysis*

To assess the area overhead and power consumption of the proposed scheme, the whole platform of each scheme is synthesized by Synopsys Design Compiler. Each scheme includes network interfaces, routers, cluster agents (for CATRA), and communication channels. For synthesis we use the UMC 90nm technology at the operating frequency of *1GHz* and supply voltage of *1V*. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme is calculated under the fft benchmark using Synopsys PrimePower in a 8×8 2D mesh. The layout area and power consumption of each platform are shown in Table 1. Comparing the area cost of the platform using CATRA with the platforms using DBAR and NoP indicates that the NoP platform consumes more power and has a higher area overhead and lower performance gain while the overhead of CATRA platform compared to DBAR is less than 1%.

V. SUMMARY AND CONCLUSION

Congestion is an important limiting factor in the performance of communication protocols in Networks-on-Chip architectures. An ideal congestion-aware routing algorithm should be able to efficiently collect, distribute, and utilize the local and non-local congestion information. In this paper, we proposed a routing algorithm that has three main characteristics. First, the congestion information of non-local nodes are gathered from the nodes that more likely are used as intermediate nodes for a given source and destination. We also ignore the congestion conditions of far distant nodes, since their status are out-dated for the current node, and thus may lead to a wrong decision in the routing unit. Moreover, the proposed method considers the congestion status of the nodes that are located in the minimal quadrant defined by source and destination. This suggests that our present method is well suited to workload isolation consolidation. Second, it is able to propagate non-local congestion information using local and distributed system without incorporating long global wires. Third idea is to utilize history-based congestion detection metric, avoiding transient changes in congestion values.

References

[1] D. Wu, B.M. Al Hashimi, M.T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection", in proceedings of ASPDAC, pp. 36–41, 2006.

[2] M. Daneshtalab, et al., "A Low-Latency and Memory-Efficient On-Chip Network," in Proceedings of 4th ACM/IEEE NOCS, 2010, France.

[3] J.C. Hu, R. Marculescu, "DyAD – smart routing for networks-on-chip", in proceedings of DAC, pp. 260-263, 2004.

[4] Intel Corporation, A touchstone delta system description, in: Intel Advanced Information, 1991.

[5] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing", in proceedings of 19th Ann. Int'l Symp. Computer Architecture, pp. 278-287, 1992.

[6] M. Li, Q. Zeng, W. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip", in proceedings of DAC, pp. 849-852, 2006.

[7] P. Lotfi-Kamran, et al., "BARP- A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs to Avoid Congestion," in Proc. of ACM/IEEE DATE, pp. 1408-1413, Mar 2008.

[8] G. M. Chiu, "The odd-even turn model for adaptive routing", IEEE Transaction on Parallel and Distributed Systems, 11:729 – 738, July 2000.

[9] A. Singh, W. J. Dally, A. K. Gupta, et al., "GOAL: A Load-Balanced Adaptive Routing Algorithm for Torus Networks", In Int. Symp. on Computer Architecture, pp. 194–205, 2003.

[10] A. Singh, et al., "Globally Adaptive Load-Balanced Routing on Tori", IEEE Computer Architecture Letters, v.3, I.1, pp.2-6, 2004.

[11] J. Duato, S. Yalamanchili, L. Ni, "Interconnection Networks: An Engineering Approach", Morgan Kaufmann, 2002.

[12] P. Lotfi-Kamran, et al., "EDXY - A low cost congestion-aware routing algorithm for network-on-chips", Journal of Systems Architecture, v.56, I.7, 2010.

[13] G. Ascia, V. Catania, M. Palesi, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip", IEEE Transaction on Computers, v.57, I.6, pp. 809 – 820, 2008.

[14] P. Gratz, , et al., "Regional Congestion Awareness for Load Balance in Networks-on-Chip", in proceedings of HPCA, pp. 203-214, 2008.

[15] S. Ma, N.E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip", in proceedings of ISCA, 2011, pp.413-424.

[16] C. Glass and L. Ni, "Maximally Fully Adaptive Routing in 2D Meshes", in proceedings of Parallel Processing, pp.101-104, 1992.

[17] S. C. Woo, M. Ohara, E. Torrie, et al., "The splash-2 programs: Characterization and methodological considerations", in proceedings of the 22nd Int. Symp. on Computer Architecture (ISCA), pp. 24–36, 1995.

[18] M. K. Martin, D. J. Sorin, B. M. Beckmann, et al. "Multifacet's general execution driven multiprocessor simulator (GEMS) toolset", SIGARCH Computer Architecture News, v. 33, No. 4, pp.92-99. November 2005.

[19] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: a 32-way multithreaded Sparc processor," IEEE Micro, vol. 25, pp. 21-29, 2005.

B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," In Proc. of the 37th annual IEEE/ACM International Symposium on MICRO, pp. 319–330, 2004.

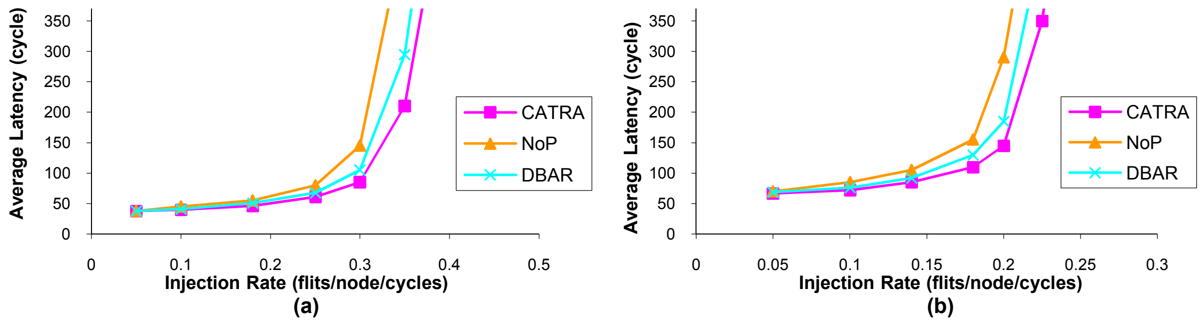


Fig. 7. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under uniform traffic model

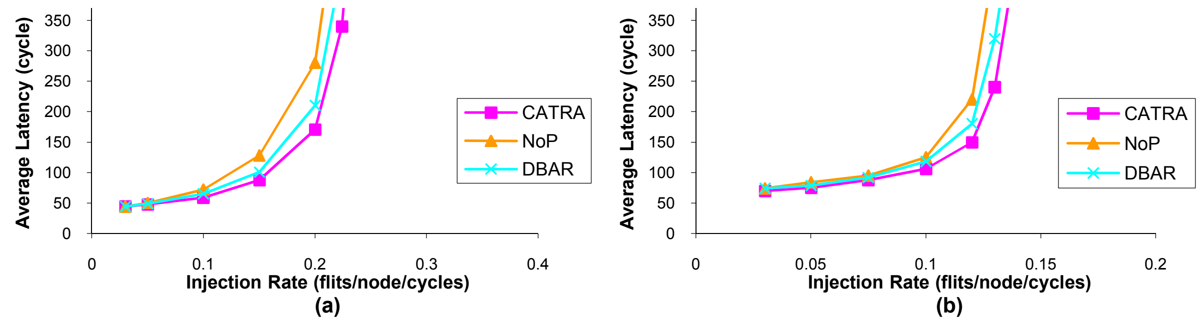


Fig. 8. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under hotspot traffic model with H=10%

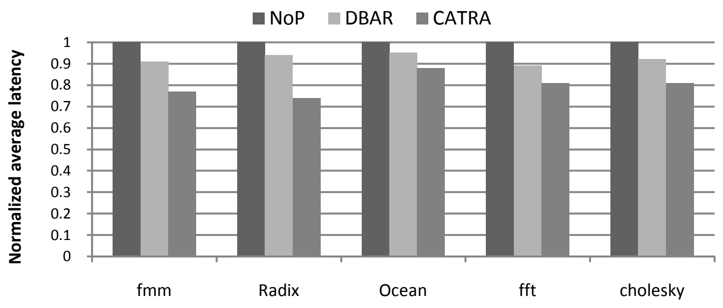


Fig. 9. Performance for application traces.