Efficient Multicast Support in Buffered Crossbars using Networks on Chip

Iria Varela Senin[1] Lotfi Mhamdi[1] Kees Goossens[1,2]
[1] Computer Engineering, Delft University of Technology, Delft, The Netherlands
[2] NXP Semiconductors, Eindhoven, The Netherlands
lotfi@ce.et.tudelft.nl, kees.goossens@nxp.com

***Abstract*—The Internet growth coupled with the variety of its services is creating an increasing need for multicast traffic support by backbone routers and packet switches. Recently, buffered crossbar (CICQ) switches have shown high potential in efficiently handling multicast traffic. However, they were unable to deliver optimal performance despite their expensive and complex crossbar fabric. This paper proposes an enhanced CICQ switching architecture suitable for multicast traffic. Instead of a dedicated internal crosspoint buffer for every input-output pair of ports, the crossbar is designed as a multi-hop Network on Chip (NoC). Designing the crossbar as a NoC offers several advantages such as low latency, internal fabric load balancing and path diversity. It also obviates the requirement of the virtual output queuing by allowing simple FIFO structure without performance degradation. We designed appropriate routing for the NoC as well as on-chip router scheduling and tested its performance under a wide range of input multicast traffic. Simulations results showed that our proposal outperforms the CICQ architecture and offers a viable architectural alternative. We also studied the effect of various parameters such as the depth of the NoC as well as the speedup requirement for high-bandwidth multicast switching.**

***Index Terms*—Multicast, Buffered crossbar fabrics.**

I. Introduction

Recent years have witnessed an increasing convergence of media services (broadcasting, cable TV and on-demand multimedia) to packet-based networks with a growing number of online multi-party applications such as online gaming, IPTV, video teleconferencing, etc. The popularity of these applications is stressing the need for efficient multicast support by Internet routers and packet-switches. Despite the various attempts to coming up with an optimal multicast switching architecture, efficiently handling multicast traffic is still an open problem.

Multicast traffic support has been studied over the last two decades. The crossbar based fabric is considered the most prominent switch interconnection architecture due to its low cost, scalability and most importantly to its intrinsic multicast capabilities [1]. Consequently, most of the research work has been carried on the input queued (IQ) crossbar-based switching architecture [2][3][4] [5][6]. Because of the multicast traffic nature, the input queueing structure represented a major design choice. Unlike unicast traffic, where a packet (cell) has only one destination output port, a multicast cell can have 1 or more destination output ports known as its fanout set. This has imposed a constraint on the number of FIFO queues per input in order to avoid the Head of Line (HoL) blocking problem. In addition to the input queueing constraint, IQ switches suffer the high inherited scheduling complexity due to the centralized control in crossbar fabrics.

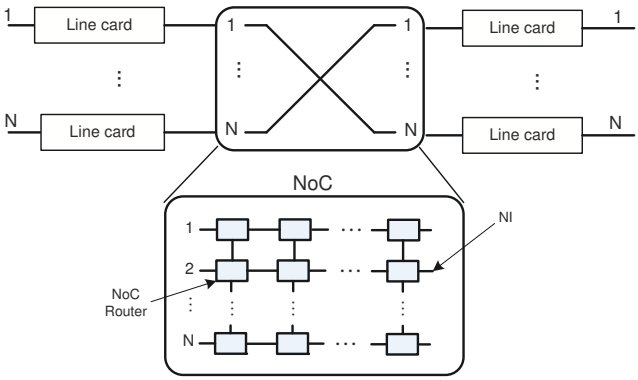


Fig. 1. The proposed NoC based crossbar fabric architecture.

The centralized scheduling complexity of unbuffered crossbars can be solved by adding a limited amount of memory in every crosspoint of the crossbar, e.g. a buffered (CICQ) crossbar switch [7]. The appeal of the CICQ switching architecture has attracted considerable research work for multicast traffic support [1][8] [9][10]. As expected, the CICQ exhibited better performance under multicast traffic than IQ switches, however it suffers throughput degradation with increased switch size [10]. Additionally, whether a crossbar switch is buffered or not, the fabric requires long point-to-point wires to interconnect the switch inputs to its outputs. This results in long delays and consumes high power to drive these wires. Another issue is the choice of the input queueing structure. Using just one FIFO per input port results in low performance due to the HoL problem, whereas avoiding the HoL is virtually impossible for multicast traffic and medium to large switch size [4].

This paper calls for rethinking buffered crossbar packet-switch design for efficient multicast traffic support. Instead of a buffered crossbar fabric with dedicated internal buffers, we design the crossbar fabric as a Network on Chip (NoC). Each internal crosspoint buffer is replaced by a NoC router. The proposed architecture consists of one FIFO queue per input port, a NoC as its interconnect stage and output ports,

as depicted in Figure 1. When a multicast packet arrives at an input port, i, it gets queued in $FIFO_i$ and waits its turn to be switched. The packet at the head of $FIFO_i$ gets switched to the first on-chip router $(R_{i,1})$ as soon as there is space in the router's input buffer. From then on, the packet gets routed through the multi-hop NoC to its set of destination output ports. This paper describes appropriate multicast routing algorithms for the proposed NoC. Multicast traffic is routed using two strategies: copy network and multicast network. In the first strategy, copies of the packet are made (as many copies as destination output ports) in the input line card and packets are treated as unicast. The second strategy makes use of the multicast capability of each NoC crossbar router.

Our design offers several advantages when compared to traditional CICQ design. First, speedup, because short wires allow reliable high-speed signalling and simple local arbitration per on-chip router. Additionally, because arbitration is distributed over the routers, the long wires from crossbar input/output ports to the central scheduler (for unbuffered crossbar) or scheduler per input-output port pair (for buffered crossbar) are eliminated. Second, load balancing, because paths from different input-output port pairs share the same router buffers. This is in contrast to traditional CICQs, where each buffer is dedicated to each input-output pair. Path diversity allows traffic from an input port to follow different paths to its destination output port. This sharing results in *load balancing of buffer space* between different flows, and possibly in a reduction of the size of the buffers.

An important feature of our proposed architecture for multicast traffic support is its simple and efficient FIFO input queuing structure. Traditional crossbar based architectures (both IQ and CICQ) suffer from the HoL blocking. To avoid this problem for multicast traffic, a costly solution known as the multicast virtual output queueing (MC-VOQ) is required [4]. Alternatively, k multicast queues per input are required, where higher values of k result in better performance but at higher costs [11]. In addition to this, multicast fanout splitting policy is often required to achieve acceptable performance [2]. However, this mandates higher crossbar frequency as compared to the input-output line speeds, to account for the multiple submissions (copies) of the same packet over the serial links between the line cards and the crossbar core. Our architecture overcomes the above problems as only one FIFO queue is maintained by input and non-fanout splitting is used. Multi-hop NoC-based crossbar fabrics do not suffer from HoL blocking because the NoC is pipelined (multi-stage). As a result, packets from a single input port, heading for different output ports, are accepted by the NoC even when some of the output ports may be blocked. Path diversity and buffer load balancing ensure that packets destined to different output ports interfere much less with each other.

The remainder of this paper is structured as follows. In Section II we provide some background knowledge on multicasting and discuss related work. In Section III we introduce our NoC-based crossbar fabric, referred to as the Unidirectional NoC (UDN). We explain its dynamics and describe two novel multicast routing algorithms for the UDN architecture. Section IV introduces a simulation-based throughput and latency analysis for different traffic types and speedup values and compares the performance to that of a traditional CICQ switch. We draw conclusions in Section V.

II. Related Work

Multicast support has been considered an obvious must for the Internet due to the advantages it offers such as optimized network performance, better use of resources, scalability, and reduced network load. Consequently, various studies have tried to develop suitable architectures for multicast traffic [3][12]. The earliest and most straightforward multicast solution was the use of copy networks. In copy networks, an input multicast packet is replicated as many times as the number of its destination output ports. Then all copies of the packet are treated as unicast packets. However, copying packets has two disadvantages: increasing the required bandwidth and making packets contend for crossbar access multiple times. For these reasons, most of the studies have then focused on the performance of crossbar switches for multicast traffic. The crossbar fabric natural multicast capability avoids copying the packet to all the destinations. It has the capability to transfer a packet simultaneously to multiple outputs using simultaneous (parallel) switching paths.

In a $N \times M$ router with multicast capabilities, a multicast packet arriving to any of the N input ports can have any set of destination output ports between 1 and M. The set of these destinations is known as the packet's fanout set. One of the popular input multicast queueing structures used was the adoption of one FIFO queue per input port of the switch [13]. However, using only one FIFO per input resulted in a severe HoL blocking problem, similar to the unicast case. Completely avoiding the HoL blocking would require to maintain up to $2^M - 1$ separate FIFO queues per input, each per distinct fanout set. This architecture, known as the multicast VOQ (MC-VOQ) switching architecture [4], is considered unfeasible for even medium sized switch due to the high number of queues required. As a result, researchers have been looking for alternative queuing structures. A compromise between maintaining one FIFO per input and using the MC-VOQ structure was the use of k FIFO queues per input $(1 \leq k \ll 2^M - 1)$ [11]. Because k is smaller than the fanout set cardinality, how to distribute incoming traffic over the k queues is important as it affects the scheduling performance. Appropriate cell placement schemes have been proposed to address this issue [11][14]. A more recent result proved that that crossbar-based switches could never reach 100% throughput for increased port numbers [10].

An equally important factor for multicast traffic handling is the multicast service discipline used. Two known service disciplines exist. The first is named non-fanout splitting policy, in which all the copies of a packet must be sent in the same packet time. If any of the output packets loses contention for an output port, none of the output packets are transmitted and the packet must try again in the next packet time. The non-fanout

splitting policy is simple to implement, however it suffers low performance. The second discipline is called fanout-splitting, in which output packets may be delivered to the output ports over any number of packet times. Only those output packets that are unsuccessful in one packet time continue to contend for output ports in the next packet time. Research has shown that fanout-splitting enables a higher switch throughput for little increase in implementation complexity [13]. However, it also requires additional bandwidth between the input line cards and the crossbar fabric core and this is undesirable.

Our work differs from previous solutions by proposing a multihop NoC fabric crossbar for multicast traffic. Our proposed architecture does not require sophisticated nor expensive input memory. It is natively designed for simple input queuing structure. One FIFO per input port is sufficient as the multi-hop NoC allows a high path diversity and low HoL blocking. This is in contrast to traditional crossbar switches, in which k FIFO queues are required. Additionally, our proposal inherently implements non-fanout splitting policy without performance degradation. This is due to the architectural property of the NoC crossbar switch, where packets are injected once in the NoC (crossbar core) without any need for input-output port synchronization, as it is the case with conventional crossbar switches.

III. The Unidirectional NoC Crossbar Fabric

The Unidirectional NoC (UDN) crossbar fabric is shown in Figure 2. The input line cards with FIFO queues are connected to the input of the crossbar and the output ports of the switch are connected to the crossbar chip outputs. As is common, our switch operates on fixed size packets (cells), where variable-length packets are segmented into fixed size cells upon entering the switch and reassembled at the exit. In the following *cells* will refer to the incoming cells arriving at the line cards, *packets* will refer to the packets in the NoC, and *routers* are part of the NoC in the crossbar fabric. Cells arrive at the input line cards and are transferred to the crossbar fabric chip when there is space in the crossbar's network-interface (NI) buffers. Cells are packetized by the ingress NIs; the routers then route these packets to the egress NIs, where the cells are depacketized and forwarded to the output line cards. The NoC is a two-dimensional mesh of packet-switched routers, with network interfaces (NI) on two opposing sides of the mesh. The mesh has N inputs and N outputs, and is scalable in the number of stages M.

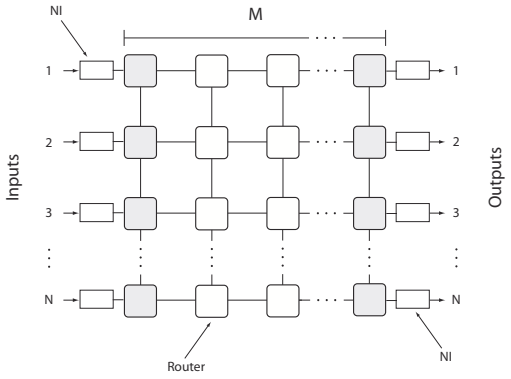


Fig. 2. The UDN crossbar fabric architecture.

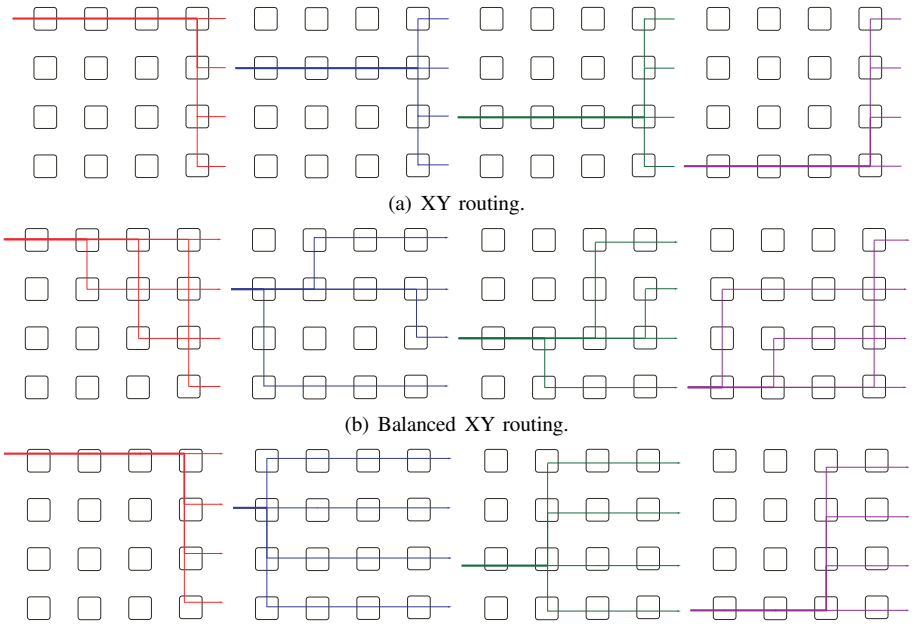
A. Multicast Switching Strategy

The set of destination output ports (fanout set) of a packet is decoded in the packet's header using a bit mask. The bit mask contains as many bits as the number of destination output ports in the switch. A bit set to "1" means that the output port indexed by that bit is a destination for the packet. It is worth noting that our switch implements the 'conventional' non-fanout splitting policy, where a packet is switched only once between the input port and the switch core. However, the switch core (the NoC routers) use fanout splitting internally, where a packet can be switched through a NoC router over multiple time slots.

In this paper, two types of strategies are tested for multicast traffic. In the first one, referred to as Copy Network, cells are replicated in the NIs and converted into unicast packets. Then, packets enter the switch core (NoC) and are distributed using a routing algorithm. In the second type of strategy, known as Multicast Network, cells are sent as multicast packets inside the switch core (NoC) without replication. Depending on the routing algorithm employed, a NoC router decides when a copy of a multicast packet has to be made, and at what stage inside the NoC. Since a multicast packet can have multiple destinations, copies of the packet may need to be generated at certain stages (coordinates) of the NoC and this is decided by the multicast routing algorithm. Each NoC router accesses the bit mask of every incoming packet and may update its corresponding bit-mask. An example of this situation is represented in Figure 4. It is a 3×3 switch in which a multicast packet, coming from input 0, enters the crossbar NoC at time slot 0 (T0). The packet has two destinations, output 0 and output 1. These destinations are marked in the bit-mask with "011". At time T1, the packet reaches Router[0][1]. This router makes a copy of the packet to send one packet to the East and another packet to the South. Each copy of the packet has an updated bit-mask ("001" for East and "010" for South). To avoid indefinite routing, when a packet gets split, its bit-mask is updated to contain only appropriate destinations.

B. Routing in the UDN

Packets are routed from the input NIs to their corresponding egress NIs. Packets follow deterministic minimal paths through the NoC, using balanced XY routing, an algorithm based on the standard XY routing. XY routing distributes the packets by first sending them in the horizontal direction to the last column and then in the vertical direction to the correct row (see Figure 3(a)). This causes an unbalanced load distribution in the mesh. To this end, we propose the balanced XY routing



(a) XY routing.

(b) Balanced XY routing.

(c) MXY routing.

Fig. 3. Example of XY, Balanced XY and MXY Routing in a 4x4 UDN.

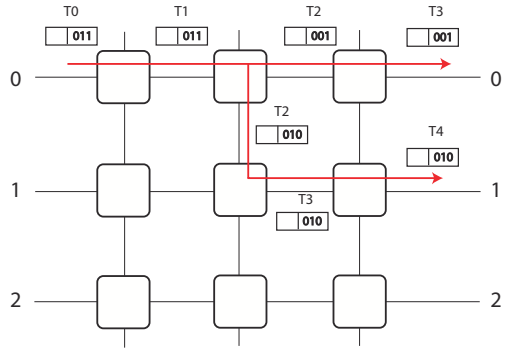


Fig. 4. Example for the bit-mask update.

algorithm. Balanced XY routing solves the unbalance problem by introducing an extra turn in one of the earlier columns. A packet that goes from input x to output y turns South/North when y *modulo* $M = (N - i + j)$ *modulo* M, and East/West when $y = j$, where i, j indicate the current router position in the mesh, N the number of inputs/outputs, and M the number of stages of the mesh. This is depicted in Figure 3(b), where each input-output pair turns in a different column. This algorithm is tested for Copy Network strategy and for Multicast strategy.

This algorithm can be simplified if all the packets belonging to the same input, take the turning decision in the same row. All the packets from input i to an output, will turn in the same column j. In this way, the number of packets that can have the maximum destination is augmented. We refer to this algorithm as the MXY algorithm. An example for this algorithm is represented in Figure 3(c): each input turns in a determined column to send the packets to the different outputs.

Algorithm 1 Balanced XY.

```
Switch (packet comes from input)
  case North:
    if i == y then go East else go South
  case South:
    if i == y then go East else go North
  case West:
    if y % M == (N-i+j) % M then
       if y > j then go South else go North
    else go East
```

Algorithm 2 MXY.

```
Switch (packet comes from input)
case North:
    if i == y then go East else go South
case South:
    if i == y then go East else go North
case West:
    if (N-1) % M == (N-i+j) % M then
    if y > j then go South else go North
    else go East
```

IV. Performance Analysis

This section presents a performance study of the proposed NoC-based crossbar and compares it to a traditional CICQ buffered crossbar that uses round-robin scheduling and N^2 internal buffers of one cell each. We tested the performance of the proposed UDN architecture by tuning its various parameters, such as switch size (N), the number of stages (columns or depth) of the NoC mesh (M), and the routing as well as the arbitration algorithm used. Different input traffic scenarios are used including Bernoulli uniform, Bursty uniform as well as non-uniform Diagonal traffic [15], where input i sends 2/3

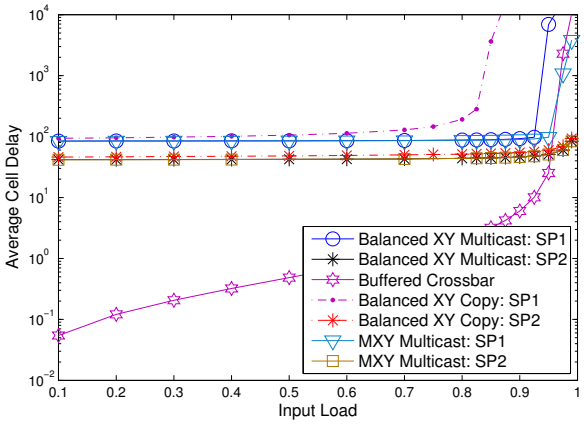


Fig. 5. A 32×32 switch under Bernoulli uniform multicast traffic.

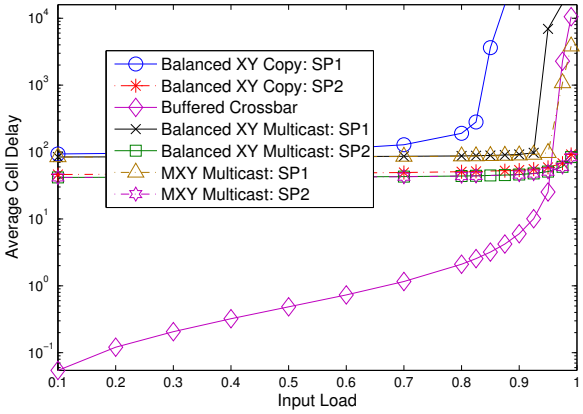


Fig. 6. A 32×32 switch under Double Diagonal multicast traffic.

and 1/3 of its load to outputs i and $i+1$, respectively.

The arrival process consists of multicast traffic flows and arrivals at different input ports are independent of each other. The fanout set of every incoming packet (set of its destination output ports) is chosen uniformly over all possible destination addresses. The fanout size follows an exponential distribution with an average of $\frac{N}{2}$, where N is the number of output ports. Simulations run for 1 million time slots and statistics are gathered when fourth of the total simulation length has elapsed. A time slot is defined as the inter-arrival time of two packets to an input, which is also equal to the time it takes a packet to go from one NoC router to another.

A. *UDN under multicast traffic*

Because our proposed UDN architecture is a multi-hop NoC, we expect it to run faster than conventional CICQ switching due to the short wires connecting the NoC routers and their size, as opposed to the long 'back-to-back' delay it takes to switch a packet across a synchronous crossbar fabric. We therefore use speedup (referred to as SP in the experiments and defined as the ratio of the crossbar NoC to the line-card frequency) in our experiments to reflect the expected high frequency of the UDN architecture. In the experiments, SP1 refers to a speedup value of 1 and SP2 refers to a speedup value of 2. Here, we compare a 32×32 UDN to a traditional CICQ switch of the same size under different traffic settings. UDN is tested for both Copy Network and Multicast Network strategies. Copy Network strategy uses only the Balanced XY algorithm while the Multicast Network strategy uses both Balanced XY and MXY algorithms. They are referred to as Balanced XY Copy, Balanced XY Multicast and MXY Multicast respectively in the graphs. Under Bernoulli uniform traffic, depicted in Figure 5, CICQ performs better under light load (<95%) when UDN employs SP2. This is attributed to the multi-hop delay of the UDN as compared to the CICQ. For high loads, UDN outperforms CICQ for SP2.

The same trend is observed under Double Diagonal traffic, illustrated in Figure 6. UDN has a similar response than with

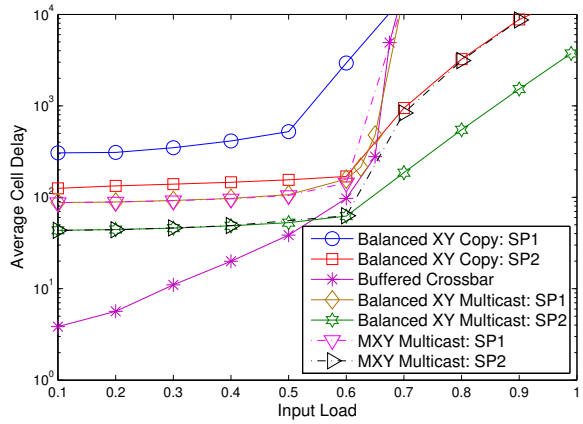
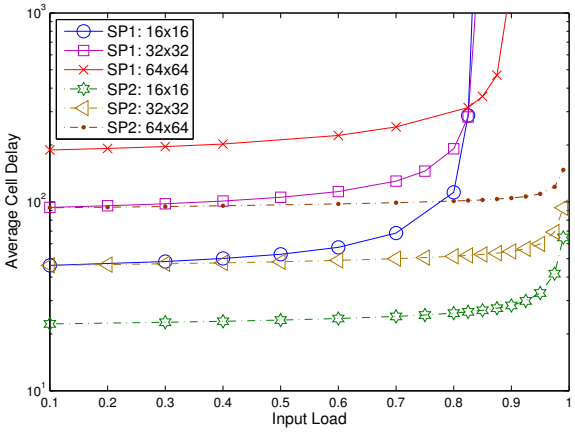


Fig. 7. A 32×32 switch under Bursty uniform multicast traffic.

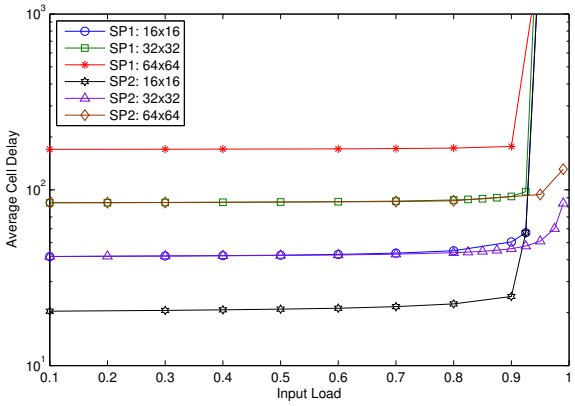
Bernoulli Uniform traffic. With speedup two and high loads (>95%), UDN outperforms CICQ, i.e. has a lower average cell delay and better throughput. Figure 7 shows both UDN and CICQ under Bursty uniform traffic, with burst size of 16 packets. As we can see, when the input traffic is bursty, the outcome is quite different. In this case, bigger sized flows cause much HoL blocking and UDN performs better than the CICQ above 60% load even with SP1. From the 3 previous Figures (Figure 5, 6 and 7) we can see that Balanced Multicast and MXY Multicast perform better than Balanced XY Copy Network strategy. This is attributed to the better network resources use (share) when employing Multicast Network strategy as fewer packets are routed through the NoC-based switch core.

B. *Varying parameters*

Figures 8 illustrates the average cell delay, in time slots, of the UDN with varying crossbar sizes N and speedup values, SP, under Bernoulli uniform traffic. The UDN architecture has a scalable delay performance for multicast traffic for increasing switch sizes, in the sense that the saturation throughput increases with increasing switch sizes, N, for both Balanced XY

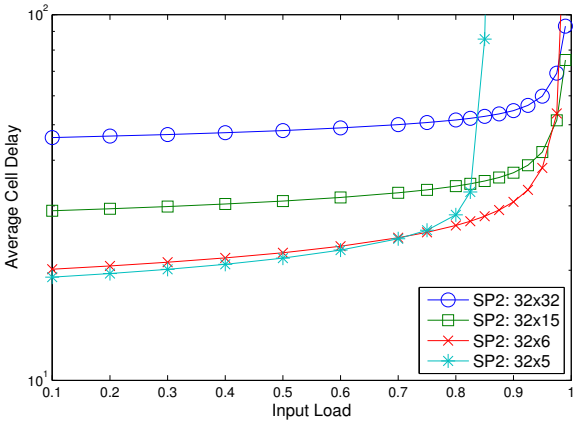


(a) Balanced XY Copy.

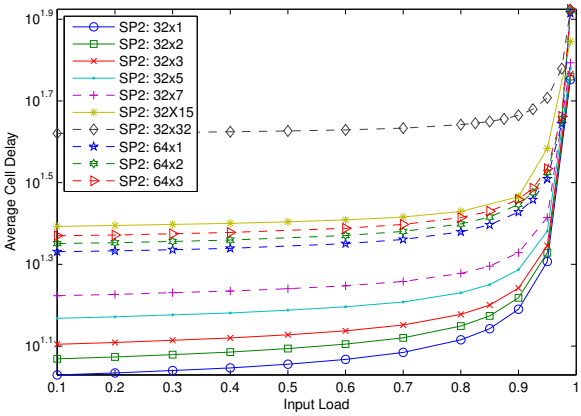


(b) Balanced XY Multicast.

Fig. 8. UDN under Bernoulli traffic for different switch sizes.



(a) Balanced XY Copy.



(b) Balanced XY Mcast.

Fig. 9. UDN under Bernoulli uniform traffic for varying M

Copy (Figure 8(a)) and Balanced XY Multicast (Figure 8(b)). Under SP2, Balanced XY Copy and Balanced XY Multicast have 100% throughput.

Varying the depth of the mesh, M, can reduce the cost of the switch while maintaining good performance. This is illustrated in Figure 9(a) where, under Balanced XY Copy, the number of columns of the switch (depth M) can be reduced by a factor of 5 while keeping full throughput. Note that the switch size used in Figure 9 is $N \times M$, where N refers to the number of switch input/output ports and M to the switch core (NoC) depth (number of stages) i.e., 32×5 refers to a UDN with 32 input/output ports and a NoC of depth 5. Balanced XY Multicast has better performance as only 1 column is sufficient to manage all the input load both for a 32×32 and a 64×64 switch. This is because the fewer columns there are, the fewer copies have to be made (see Figure 9(b)). When there is more than 1 column, packets turn in different positions for different output ports. Figure 3(b) shows how i.e. a packet going from input 0 to output 1 meets the path of the packets going from input 0 to output 2 in the NoC Router[1][2]. Then, two different copies of the same original packet may contend for the output. On the other hand, when there is only 1 column, the packet would not have been split avoiding the possible contention and minimizing the number of packets. Under SP1, no optimization can be made as even with maximum M, the throughput is not 100% for full load (Figure 8(a), 8(b)).

Round robin was the scheduling algorithm performed in the previous tests. More sophisticated arbiters can be implemented i.e. choosing the packet having the maximum fanout. Figure 10 shows this analysis. Since the NoC routers have few number of ports, the scheduling scheme seems to be not critical.

We can withdraw a number of conclusions out of this analysis. We can see that Multicast Networks outperforms Copy Networks when we compare Balanced XY Copy with Balanced XY Multicast for different traffic scenarios. It is worth noting that the UDN outperforms Buffered Crossbar (CICQ) under multicast traffic scenarios and speedup two (SP2). Also, the number of columns M can be reduced by a factor of 5 for SP2 when using Copy Network, and to only 1 column for SP2 when we employ Multicast Network strategy. This results in significant cost saving as compared to a fully buffered crossbar.

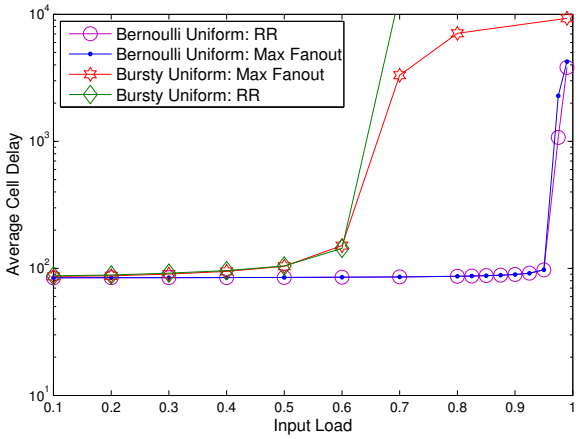


Fig. 10. MXY Multicast under different scheduling algorithms.

V. Conclusions

In this paper we proposed a novel architecture for a switch fabric under multicast traffic. We described a multi-hop crossbar fabric based on NoC instead of the dedicated crosspoint-based buffered crossbar switch. Our proposal offers several architectural advantages when compared to a CICQ switch, such as speedup, path diversity and load balancing. With regards to multicast traffic, the UDN proposal has two main advantages of using a single FIFO per input and adopting non-fanout splitting policy while maintaining high-performance. This is in contrast to traditional buffered crossbars that maintains k queues per input and require costly fanout-splitting policy. The UDN architecture exhibits good performance under different traffic patterns and is scalable in terms of switch port count and speed per port.

References

[1] L. Mhamdi, G. N. Gaydadjiev, and S. Vassiliadis, "Efficient Multicast Support in High-Speed Packet Switches," *Journal of Networks*, vol. 02, no. 03, pp. 28–35, June 2007.

[2] N. McKeown, "A Fast Switched Backplane For A Gigabit Switched Router," *Business Commun. Rev.*, vol. 27, no. 12, 1997.

[3] M. Guo and R. Chang, "Multicast ATM Switches: Survey and Performance Evaluation," *Computer Communication Review*, vol. 28, no. 02, pp. 98–131, April 1998.

[4] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 465–477, 2003.

[5] D. Pan and Y. Yang, "FIFO-Based Multicast Scheduling Algorithm for Virtual Output Queued Packet Switches," *IEEE Trans. Comput.*, vol. 54, no. 10, pp. 1283–1297, 2005.

[6] Andrea Bianco, Paolo Giaccone, Chiara Piglione, Sonia Sessa, "Practical Algorithms for Multicast Support in Input Queued Switches ," *IEEE, High Performance Switching and Routing*, June 2006.

[7] M. Nabeshima, "Performance Evaluation of Combined Input-and Crosspoint-Queued Switch," *IEICE Transactions On Communications*, vol. B83-B, no. 3, March 2000.

[8] L. Mhamdi and M. Hamdi, "Scheduling Multicast Traffic in Internally Buffered Crossbar Switches," *IEEE International Conference on Communications (ICC)*, pp. 1103–1107, June 2004.

[9] S. Sun, S. He, Y. Zheng, and W. Gao, "Multicast Scheduling in Buffered Crossbar Switches with Multiple Input Queues," *IEEE Workshop on High Performance Switching and Routing (HPSR)*, pp. 73–77, May 2005.

[10] P. Giaccone and E. Leonardi, "Asymptotic Performance Limits of Switches with Buffered Crossbars," *IEEE Transactions on Information Theory*, vol. 54, no. 02, pp. 595–607, Feb. 2008.

[11] S. Gupta and A. Aziz, "Multicast Scheduling for Switches with Multiple Input-Queues," *Proceedings of Hot Interconnects*, pp. 28–33, 2002.

[12] X. Liu and H. T. Moufath, "Overflow control in multicast networks," *Proc. of Canadian Conf. on Eleectrical and computer Engineering*, pp. 542–545, 1993.

[13] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast Scheduling for Input-Queued Switches," *IEEE Journal in Selected Areas in Communications (JSAC)*, vol. 15, pp. 855–866, June 1997.

[14] L. Mhamdi, "On the Integration of Unicast and Multicast Cell Scheduling in Buffered Crossbar Switches," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 50, no. 02, pp. 818–830, June 2009.

[15] P. Giaccone, D. Shah, and B. Prabhakar, "An implementable Paraller Scheduler for Input-Queued Switches," *IEEE Micro*, vol. 19, no. 01, pp. 1090–1096, January/February 1999.