

# Preventing Out-of-Sequence for Multicast Input-Queued Space-Memory-Memory Clos-Network

**Yu, Hao; Ruepp, Sarah Renée; Berger, Michael Stübert; Dittmann, Lars**

*Published in:*
Proceedings of OPNETWORK 2011

*Publication date:*
2011

# Preventing Out-of-Sequence for Multicast Input-Queued Space-Memory-Memory Clos-Network

**Hao Yu, Sarah Ruepp, Michael S. Berger, and Lars Dittmann**
*Technical University of Denmark*
*2800 Kgs. Lyngby, Denmark*
*E-mail: {haoyu, srru, msbe, ladit}@fotonik.dtu.dk*

**Abstract**
This paper proposes an out-of-sequence (OOS) preventative cell dispatching algorithm, the multicast flow-based round robin (MFRR), for multicast input-queued space-memory-memory (IQ-SMM) Clos-network architecture. Independently treating each incoming cell, such as the desynchronized static round robin (DSRR), can lead to cells of the same packet being disordered after traversing the switch fabric. Extra reassembly buffer and delay are thus required at the outputs of the switch fabric to reorder the cells. Simulation results obtained by OPNET Modeler show that the MFRR can eliminate such OOS problem by monitoring the changes of received fan-out vectors.

## 1. Introduction

Multicast enables networks to handle traffic in a resource-efficient manner by introducing some complexity to the network nodes. By the capability of copying and sending the same packet to different destination ports, network nodes can reduce the traffic load. This benefit is especially significant regarding some high-bandwidth applications, such as Internet Protocol Television (IPTV), video broadcasting, and videoconferencing.

A number of publications attempt to build high-speed and large-scale switches [1]-[11]. In terms of the types of switch fabric, they are usually categorized into single-stage switches and multi-stage switches. A crossbar switch, as shown in Fig. 1, is a single-stage switch fabric that has two attractive properties: simple architecture and internally non-blocking. However, such a fabric architecture has a limited scalability because the number of cross points increases in a square proportion ($N^2$) to the switch size, i.e. the number of input/output ports.

As the port speed increases to today's 100 Gbit/s, the scalability of switches becomes an important issue and factor to the switch architecture design. The low scalability of crossbar switch fabrics makes them unattractive to large-scale switches. On the other hand, using multi-stage switch fabrics to build high-capacity switches has become very common [12]. Clos-network, as shown in Fig. 2, is a multi-stage switch fabric that is considered more scalable and cost-efficient compared to the crossbar switch fabric. A Clos-network consists of three stages, i.e. the input stage, the central stage, and the output stage. Crossbar switching elements (X-SEs) are placed at each stage to construct the Clos-network, and are referred to as the input module (IM), the central module (CM), and the output module (OM) based on their location.

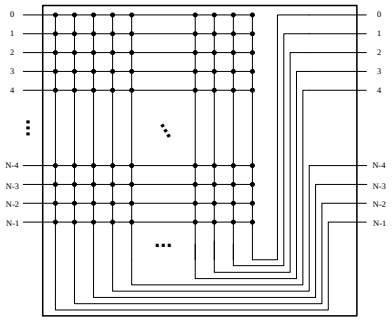


Figure 1: A crossbar switch fabric architecture.

Various scheduling algorithms have been developed for different Clos-network architectures regarding buffer allocations. The architecture where memories are only allocated in the IMs/OMs is referred to as the memory-space-memory (MSM) architecture. CRRD/CMSD [7] and MWMD [8] have been proposed for such an architecture using round-robin arbitration. One disadvantage of these schemes is that the IMs and OMs use shared memories, resulting in a requirement of a speedup to the buffer, which hinders the scalability of the switch. To solve this problem, a bufferless space-space-space ($S^3$) Clos-network architecture is proposed in [6]. Based on the static round-robin dispatching (SRRD), Distro [6] is proposed for the $S^3$ architecture and is demonstrated to achieve 100% throughout under uniform traffic. Due to that no buffers are allocated at the CMs, MSM and $S^3$ architectures require complex algorithms to solve the cell contention problems to avoid blocking and are thus poorly scalable. The fully buffered memory-memory-memory (MMM) architecture, despite having no cell contention problem, is not cost-effective in hardware implementation. Then in [9], a space-memory-memory (SMM) Clos-network architecture with desynchronized static round-robin (DSRR) dispatching scheme is proposed. The study demonstrates that SMM can achieve 100% throughput with DSRR under admissible traffic. However, the SMM with DSRR in [9] uses output-queued SMM (OQ-SMM) scheme for the buffered stages, where speedup is again required. In addition, since bufferes are placed in the central stages, cells can experience different delays and thus resequencing is required to solve the out-of-sequence (OOS)

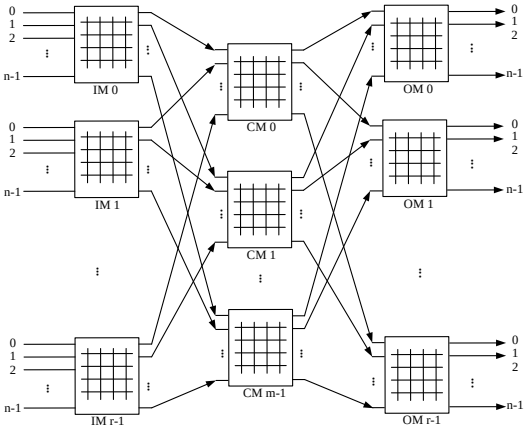


Figure 2: A Clos-network switch fabric architecture

problem at the output. When concerning multicast, DSRR, initially designed for unicast and not evaluated for multicast, can result in a serious OOS problem.

In order to reduce the requirement of the speedup of the OQ-SMM and solve the OOS problem caused by DSRR, this paper proposes an input-queued SMM (IQ-SMM) architecture and two cell dispatching algorithms: multicast flow-based DSRR (MF-DSRR) and multicast flow-based round-robin (MFRR). Input-queued crossbar switching elements (IQ-X-SEs) are leveraged as the CMs and OMs in this architecture. The MF-DSRR or MFRR is independently run in the IMs, distributing incoming cells to the CMs. MF-DSRR utilizes the low-complexity connection changing pattern of the DSRR. MFRR is able to eliminate the in-packet OOS problem and reduces the reassembly buffer size, and thus results in smaller reassembly delay.

The rest of the paper is organized as follows. In Section 2, the architecture of the system and some notations used throughout this paper are presented. In Section 3, the descriptions of the proposed multicast scheduling algorithm are presented. In Section 3, the OPNET models of the switch architecture are illustrated. In Section 4, the performance of the proposed cell dispatching algorithm is evaluated by simulations carried out in OPNET Modeler. Finally the paper is concluded in Section 5.

## 2. Input-Queued Space-Memory-Memory Clos-Network

We assume that a first-in-first-out (FIFO) queue is installed at each input port to temporally store multicast packets. Packets are assumed to be segmented into fixed-size cells in the input port processors (IPPs) before entering into IMs and reassembled at the output port processors (OPPs) after traversing the switch fabric. The IQ-SMM Clos-network switch model consists of three stages and is denoted as $C(n,m,r)$. As shown in Fig. 3, the switch has $r$ IMs of size $n \times m$, $r$ OMs of size $m \times n$, $m$ CMs of size $r \times r$. Each IM/OM has $n$ connections to IPPs/OPPs, and $m$ interstage connections to CMs. Only one interstage connection exists between an IM/OM and a CM. The number of input/output ports of the switch is $N = nr$.

IMs forward incoming cells to the CMs following certain cell dispatching (CD). Each CM/OM has $r/m$ FIFO input queues, each of which is connected to one interstage link. Several notations used throughout this paper are listed as follows:

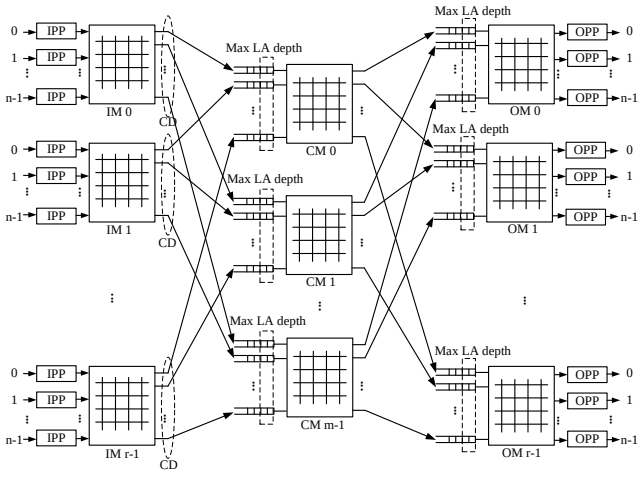

Figure 3: Input-queued space-memory-memory Clos-network

| | |
|---|---|
| $IM_i$ | $i^{th}$ *input module, where* $0 \le i \le r-1$ |
| $CM_k$ | $k^{th}$ *central module, where* $0 \le k \le m-1$ |
| $OM_j$ | $j^{th}$ *output module, where* $0 \le j \le r-1$ |
| $I_{i,p}$ | $p^{th}$ *input of* $IM_i$, *where* $0 \le p \le n-1$ |
| $O_{j,q}$ | $q^{th}$ *output of* $OM_j$, *where* $0 \le q \le n-1$ |
| $Q_{C\,i,k}$ | $i^{th}$ *input queue of* $CM_k$ *connected to* $IM_i$ |
| $Q_{O\,k,j}$ | $k^{th}$ *input queue of* $OM_j$ *connected to* $CM_k$ |
| $IL_{i,k}$ | *the interstage link connecting* $IM_i$ *and* $CM_k$ |
| $CL_{k,j}$ | *the interstage link connecting* $CM_k$ *and* $OM_j$ |

It is assumed that each packet carries a fan-out vector $\boldsymbol{b} = \langle b_j \rangle, b_j \in \{0,1\}, 0 \le j \le N-1$, where $b_j = 1$ indicates the packet is destined to the $j^{th}$ output port, otherwise $b_j = 0$. Cells generated from the same packet have the same fan-out vectors. Cells are replicated as far downstream as possible in the switch model. A bit-cluster is defined as a set of $n$ consecutive bits in the fan-out vector, and is denoted as $\boldsymbol{c}_d = \langle b_j \rangle, nd \le j \le nd + n - 1$. The intersection of different bit-clusters is empty, $\boldsymbol{c}_{d_1} \cap \boldsymbol{c}_{d_2} = \emptyset, if\ d_1 \ne d_2$. Therefore the fan-out vector can be expressed by bit-clusters, $\boldsymbol{b} = \langle \boldsymbol{c}_d \rangle, 0 \le d \le r-1$. We define $|\boldsymbol{c}_d| = \min\left(1, \sum b_j\right)$. $CM_k$ multicasts the incoming cell by examining $r$ bit-clusters of the fan-out vector. If $\boldsymbol{c}_d \ne 0$, $CM_k$ sends a copy of the cell to $OM_d$. $OM_d$ examines all the bits in $\boldsymbol{c}_{d-1}$, and if $b_j = 1$, $OM_d$ sends a copy of the cell to $O_{d,j-nd}$.

## 3. Multicast Flow-based DSRR (MF-DSRR) and Multicast Flow-based Round Robin (MFRR)

### 3.1 MF-DSRR Dispatching

As shown in Fig. 4, the IM connection pattern of DSRR changes after each cell time, resulting in a balanced distribution of cells to the CMs and low-complexity connection setup process. However, when DSRR is applied to the IQ-SMM architecture in Fig. 3, it causes a serious OOS problem.

MF-DSRR utilizes the low-complexity connection setup of the DSRR and modifies connection pattern to that instead of being changed after each cell time, it is changed *when and only when* a notification of a fan-out vector change is received from the input. More specifically, when any new fan-out vector is detected, regardless on which input port the change is detected, each input of $IM_i$ moves its connection to the next interstage link of $IM_i$ in a round-robin manner before a time slot begins. At most one change of the connection pattern is performed before a time slot. The example shown in Fig. 5 demonstrates how the connection pattern changes for a 4×6 IM. The initial configuration can be ($I_{i,0} \to IL_{i,0}$, $I_{i,1} \to IL_{i,1}$, $I_{i,2} \to IL_{i,2}$, $I_{i,3} \to IL_{i,3}$) as shown in the first graph. As a fan-out change occurs on $I_{i,0}$ from $f_0$ to $f_0'$, each input maps its connection to the next interstage link, resulting in a connection as ($I_{i,0} \to IL_{i,1}$, $I_{i,1} \to IL_{i,2}$, $I_{i,2} \to IL_{i,3}$, $I_{i,3} \to IL_{i,4}$). This connection pattern may be kept for several cell times until another change is detected in the third graph, where $I_{i,3}$ has a fan-out change from $f_3$ to $f_3'$. Then the connection pattern becomes ($I_{i,0} \to IL_{i,2}$, $I_{i,1} \to IL_{i,3}$, $I_{i,2} \to IL_{i,4}$, $I_{i,3} \to IL_{i,5}$).

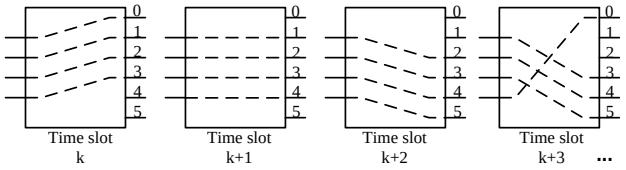


**Figure 4. Desynchronized static round robin connection pattern** (DSRR)

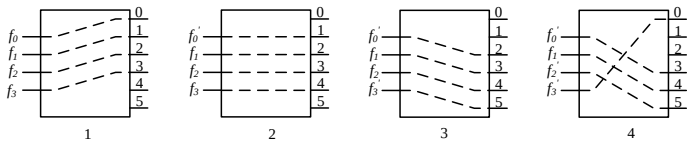


**Figure 5. Multicast flow-based DSRR (MF-DSRR)**

### 3.2 MFRR Dispatching

Independently run in each IM, MFRR also requires that each input monitors the change of received fan-out vectors. Even though MF-DSRR has a low connection setup complexity, it can cause cells carrying the same fan-out vector to be distributed to different CMs because when a fan-out change occurs on one input, the whole connection of the IM is changed. During a transmission of a flow of cells of the same fan-out vector, it is possible that the connection pattern is changed due to a fan-out change detected on other input. This will cause transmitting these cells to different CMs. In MFRR, each input maintains its own connection and changes to an idle interstage link when the input detects a fan-out change on it.

Since each input sets up the connection independently, the problem of which idle interstage link should be chosen to set up the new connection needs to be solved. A list called *AvailableList* is created for each IM to record the idle interstage links as its elements. Elements can be only popped from the top of the list and inserted to the bottom. When an element is popped, all the elements in the *AvailableList* are moved one position up to the top. Since there are $n$ inputs and $m$ interstage links in IM, the number of elements of the *AvailableList* is $(m-n)$. Therefore in order to use MFRR, the switch must have $m > n$. Usually for a strictly non-blocking Clos-network, $m \geq 2n-1$. Therefore, the length of the *AvailableList* in this case will be at least $n-1$.

Upon detecting a change of fan-out vector on $I_{i,p}$, $IM_i$ pops an idle interstage link $IL_{i,k'}$ from the *AvailableList*, disconnects the connection $(I_{i,p} \rightarrow IL_{i,k})$ and sets up a new connection $(I_{i,p} \rightarrow IL_{i,k'})$ instead. The released interstage link $IL_{i,k}$ is inserted to the bottom of the *AvailableList*. While establishing the new connection for the input, $IM_i$ keeps others unchanged. If $IM_i$ detects multiple changes of fan-out vector on several inputs in the same time slot, ties are broken randomly. To better demonstrate, we consider a 4×6 IM as shown in Fig. 6. The connection pattern can initially be $(I_{i,0} \rightarrow IL_{i,0}, I_{i,1} \rightarrow IL_{i,1}, I_{i,2} \rightarrow IL_{i,2}, I_{i,3} \rightarrow IL_{i,3})$ with $\{IL_{i,4}, IL_{i,5}\}$ in the *AvailableList*. Assume a new fan-out vector $f_1'$ occurs on $I_{i,1}$, $IL_{i,5}$ is popped from the list and a new configuration is established as $(I_{i,0} \rightarrow IL_{i,4}, I_{i,1} \rightarrow IL_{i,1}, I_{i,2} \rightarrow IL_{i,2}, I_{i,3} \rightarrow IL_{i,3})$. The released interstage link $IL_{i,1}$ is inserted to the bottom of the *AvailableList*, which becomes $\{IL_{i,5}, IL_{i,0}\}$. When a new fan-out vector is detected by $I_{i,3}$, $I_{i,6}$ is popped from the *AvailableList* and a connection pattern $(I_{i,0} \rightarrow IL_{i,4}, I_{i,1} \rightarrow IL_{i,1}, I_{i,2} \rightarrow IL_{i,2}, I_{i,3} \rightarrow IL_{i,5})$ is established with the list being $\{IL_{i,0}, IL_{i,3}\}$.

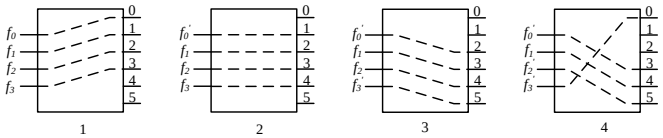


**Figure 6. Multicast flow-based DSRR (MF-DSRR)**

## 4. OPNET Models of the Input-Queued Space-Memory-Memory (IQ-SMM) Clos-Network

We implement the IQ-SMM Clos-network and the two proposed cell dispatching schemes in OPNET Modeler [14]. Variable-length packets are segmented into fixed-size cells in the IPPs and are forwarded to the IMs. The MF-DSRR and MFRR are performed by the IMs to distribute incoming cells to CMs accordingly. Each CM/OM runs as an $r \times r$ crossbar input-queued switch with the multicast scheduling algorithm described in [13].

### 4.1 Input Packet Processor (IPP) Process Model

The process model of the Input Packet Processor (IPP) is shown in Fig. 8. Parameters used in the process are initialized in the "Init" state.

After the initialization phase, the module goes to the "Idle" state and waits for further events. Packet arrivals trigger the module to execute the function InsertPacket and PacketToSegQ, which inserts the packet into a FIFO buffe and then segments it into a number of cells as the buffer was empty before this packet arrival. Then the process transit to the "Segmenting" state.

In the "Segmenting" state, PacketArrival triggers the function InsertPacket since the current packet is not finished with the segmentation process and any incoming packet is stored in the FIFO buffer. A timer is maintained by the process to calculate the cell time, which is equal to the cell size divided by the link speed. When the timer expires, the event TimeToSendACell happens and the function SendOneSegment is executed. When the current packet has no more segments, NoRemainSegments triggers the process to transit either to "Segmenting" or "Idle", depending on the buffer length. If the buffer length is greater

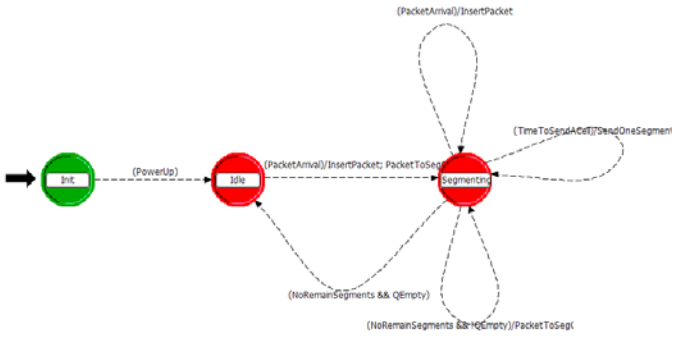


**Figure 8. Input Packet Processor Process**

than 0, indicating that there are packets waiting to be segmented, the process executes the function PacketToSegQ. Else, it transits "Idle".

### 4.2 Output Packet Processor (OPP) Process Model

The process model of the Output Packet Processor (OPP) is shown in Fig. 9. Parameters used in the process are initialized in the "Init" state. After the initialization phase, the module goes to the "Idle" state and waits for further events.

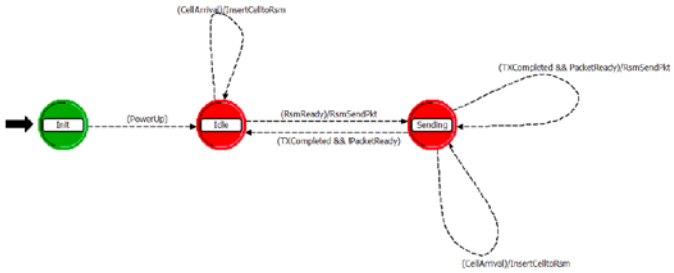


Figure 9. Output Packet Processor Process

In "Idle" state, CellArrival triggers the process to execute InsertCelltoRsm function, which inserts the incoming cell to the reassembly buffer. When a packet is reassembled, event RsmReady transits the state from "Idle" to "Sending" and executes the function RsmSendPkt.

In "Sending" state, when the packet transmission timer expires, the event TXCompleted triggers the state transition to either "Idle" or "Sending" based on whether there is any packet ready in the reassembly buffer. If there is a packet ready for transmission, RsmSendPkt is called and the process stays in "Sending". Else, the process transits from "Sending" to "Idle". During the stay in "Sending", CellArrival triggers the function InsertCelltoRsm and transition back to "Sending" because the process is busy with transmitting a packet and any incoming cells should be stored in the reassembly buffer.

### 4.3 Input Module (IM) Process Model

The process model of the Input Module (IM) is shown in Fig. 10. Parameters used in the process are initialized in the "Init" state. After the initialization phase, the module goes to the "Idle" state and waits for further events.

In "Idle", CellArrival executes the function ForwardCell and transits back to "Idle" state. The function ForwardCell reads parameters from the node and distributes the incoming cells based on the selection of the cell dispatching schemes.

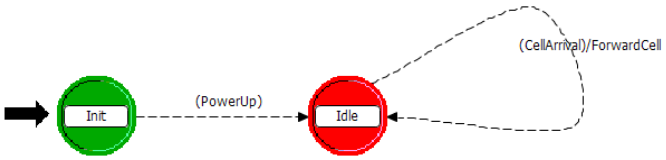


Figure 10. Input Module Process

### 4.4 Central Module/ Output Module (CM/OM) Process Model

The process model of the Central Module/Output Module (CM/OM) is shown in Fig. 11. Parameters used in the process are initialized in the "Init" state. Afterwards, the module goes to the "Idle" state and waits for further events. A periodic timer is maintained by the process to model the cell time.

In "Idle" state, CellArrival event executes the function InsertCell that inserts the incoming cells to one of the FIFO queues installed at the inputs of the CM. As the timer that emulates the cell time expires, event NextTimeSlot happens and makes a transition from "Idle" to "Request", beginning the multicast scheduling process described in [14].

Basically, the CM and the OM process run the same scheduling algorithm. The only difference is that they observe different parts of the fan-out vector. As described previously, the CM only observes the bit-clusters and the fan-out observed will be different from the actual one. For instance, if a cell has a fan-out vector $< 11000001 >$ and $n = 2$ (two outputs on the OM), then the fan-out observed by the CM will become $< 1001 >$. The OM, on the contrary, only observes the bit-cluster that has the same index as it. For instance, $OM_0$ only checks the $< 11 >$ part because it is these two bits that matter to its outputs.

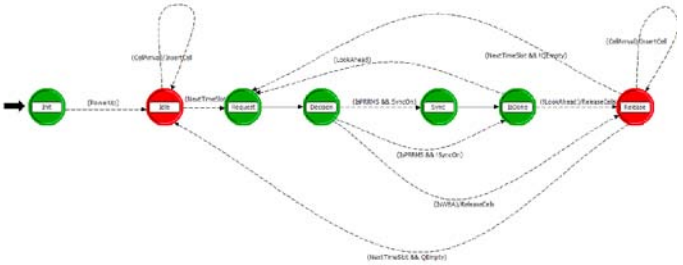

Figure 11. Central Module/Output Module Process

## 5. Simulation Results and Analysis

The simulation is carried out in OPNET Modeler [14]. DSRR is used as a reference to the MF-DSRR and MFRR.

OOS problem is categorized into two types: *inter-packet* OOS and *in-packet* OOS. Inter-packet OOS means that cells generated by different multicast packets are disordered, and in-packet OOS implies that cells generated by the same multicast packet are disordered. The reason of such a subdivision is to differentiate the characteristics of OOS and further analyze and compare different cell dispatching schemes so that appropriate reassembly methods can be applied if necessary.

Admissible traffic with average fan-out of 8 and average number of cells per packet of 13 is generated independently at each input of the simulated $C(4,7,4)$ IQ-SMM Clos-network switch. The multicast scheduling algorithms with sync to reduce multiple transmissions of cells used in CMs and OMs are described in [13]. A multicast cell may be served several times before it is removed from the queue. Unnecessary multiple transmissions can cause increased cell delays in a multicast switch. The sync mechanism aims at reducing the number of transmissions per

multicast cells while maintaining the output port utilization. In order to reduce the HOL blocking problem, both CMs and OMs can look ahead into their FIFO queues for cells that can be served to idle outputs. Since it is impractical to search an infinite depth into the queues, a maximum look-ahead depth, $A$ , is defined. If the $LA$ is reached and the IQ-SE still has idle outputs, it stops the searching and completes the scheduling process.

Fig. 12 compares the inter-packet OOS for different cell dispatching schemes. With the sync mechanism, the inter-packet OOS is reduced for both MFRR and MF-DSRR. Under high offered load, MFRR and MF-DSRR can result in more than 50% of the received cells being inter-packet OOS cells.

Fig. 13 compares the in-packet OOS for different cell dispatching schemes. MFRR schemes outperform the others with zero in-packet OOS cells. Both MF-DSRR schemes have less than 10% of total received cells being OOS cells under high offered load. DSRR schemes result in serious in-packet OOS problems. With the sync mechanism, DSRR causes more in-packet OOS cells because same-packet cells are treated independently and distributed to different CMs, resulting in a well distributed fan-out vectors in the input queues of CMs and OMs. Thus the same-packet cells have a higher probability to be disordered due to the round-robin working mechanism of the sync. A decrease of the in-packet OOS for DSRR and MF-DSRR schemes can be observed under the loads higher than 0.7. This is due to the fact that the CM queue length begins to increase non-linearly, resulting in more inter-packet OOS cells. Since the inter-packet and the in-packet OOS are considered separately, the percentage of in-packet OOS therefore decreases.

Fig. 14 shows the total number of OOS cells, the sum of inter-packet and in-packet OOS cells, for different schemes. MFRR with sync significantly reduces the OOS problem under high offered loads, while DSRR schemes cause nearly linear growths of the OOS cells with the increase of the offered load.

Fig. 15, Fig. 16 and Fig. 17 compares the inter-packet OOS in-packet OOS and total OOS cells, respectively, under different $LA$ values for the sync mechanism. With larger $LA$ values, a decrease on inter-packet OOS for each cell dispatching scheme is observed in Fig. 15. DSRR with $LA = 2$ outperforms the other two schemes. In Fig. 16, the look-ahead mechanism greatly reduces the in-packet OOS for the DSRR scheme but DSRR with $LA = 2$ still suffers from approximately 50% of all the received cells being in-packet OOS under high load. MFRR always maintains zero in-packet OOS under different $LA$ values. In terms of the total OOS cells, MFRR with $LA = 2$ outperforms the others, as shown in Fig. 17. This is because with the capability of look-ahead, the IQ-X-SEs are able to send some of the delayed cells that cause the OOS problem.

## Conclusion

This paper presented two out-of-sequence preventative cell dispatching schemes for the input-queued space-memory-memory (IQ-SMM) Clos-network architecture. MF-DSRR utilizes the connection pattern of DSRR and obtains a low implementation complexity. MF-DSRR can reduce the OOS problem but still suffers from the in-packet OOS problem due to its simplicity. On the other hand, MFRR is able to eliminate the in-packet OOS problem. With the use of a list to record the idle outputs, the MFRR can achieve a low complexity for connection setup.

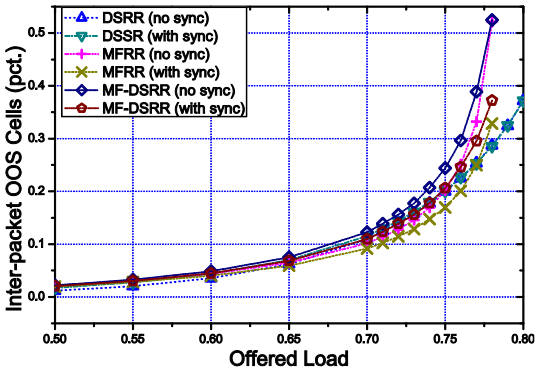


Figure 12. Inter-Packet OOS Cells, LA = 0

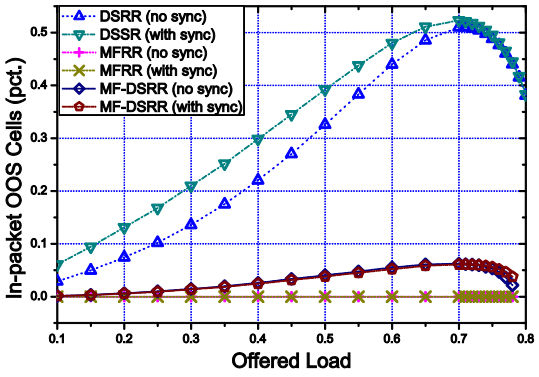


Figure 13. In-Packet OOS Cells, LA = 0

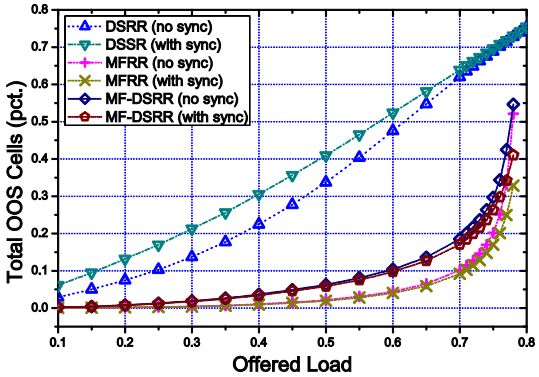


Figure 14. Total OOS Cells, LA = 0

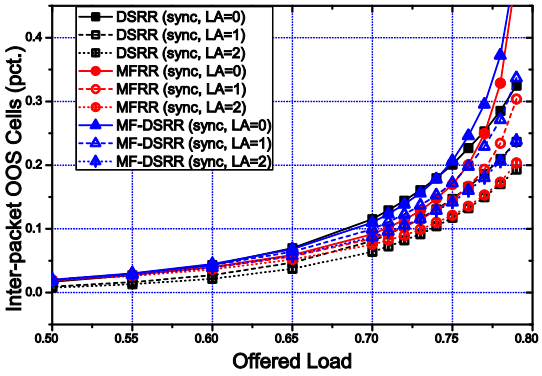


Figure 15. Inter-Packet OOS Cells, LA = 0,1,2

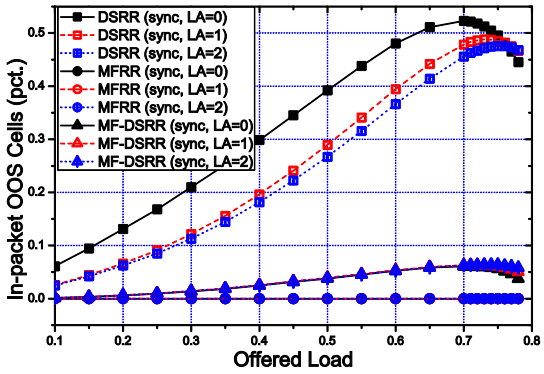


Figure 16. In-Packet OOS Cells, LA = 0,1,2

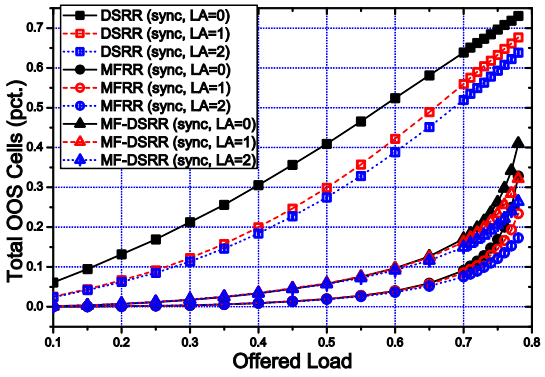


Figure 17. Total OOS Cells, LA = 0,1,2

Simulation results show that the MFRR cell dispatching outperforms DSRR and MF-DSRR in terms of reducing the OOS problem. The sync mechanism is able to improve the performance of MFRR and MF-DSRR but worsens the performance of DSRR. With the look-ahead mechanisms, the IQ-SMM architecture can further reduce the OOS problem.

## Acknowledgement

This work was supported in part by the Danish National Advanced Technology Foundation in the project *The Road to 100 Gigabit Ethernet*.

## Reference

[1] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches", *IEEE Journal on Selected Areas in Commun.*, vol. 15, no. 5, Jun. 1997

[2] D. Pan, and Y. Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches", *IEEE Trans. Comput.*, vol. 54, no. 10, Oct. 2005.

[3] D. Pan, and Y. Yang, "Bandwidth guaranteed multicast scheduling for virtual output queued packet switches", *J. Parallel Distrib. Comput.*, vol. 69, issue. 12, pp. 939-949, 2009.

[4] S. Sun, S. He, Y. Zheng, and W. Gao, "Multicast scheduling in buffered crossbar switches with multiple input queues", in *Proc. HPSR*, 2005.

[5] 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. Networking*, vol. 11, no. 3, Jun. 2003

[6] K. Pun and M. Hamdi, "Distro: A distributed static round-robin scheduling algorithm for bufferless Clos-network switches", in *Proc. GLOBECOM '02*, pp. 2298-2302, vol. 3, Nov. 2002

[7] E. Oki, Z. Jing, R. Rojas-Cessa, and J. Chao, "Concurrent round-robin-based dispatching schemes for Clos-network switches", *IEEE Trans. On Networking*, pp. 830-844, vol. 10, issue 6, Dec. 2002.

[8] R. Rojas-Cessa, E. Oki, and J. Chao, "Maximum weight matching dispatching scheme in buffered Clos-Network packet switches", in *Proc. ICC '04*, pp. 1075-1079, vol. 2, Jun. 2004.

[9] X. Li, Z. Zhou, and M. Hamdi, "Space-memory-memory architecture for CLOS-network packet switches", in *Proc. IEEE International Conference on Communications*, vol. 2, pp. 1031-1035, 2005

[10] Y. Yang, and G. M. Masson, "The necessary conditions for Clos-type nonblocking multicast networks", *IEEE Transactions on Computers*. vol. 48, no. 11, Nov. 1999.

[11] Y. Yang and J. Wang, "On blocking probability of multicast networks", *IEEE Transactions on Communications*, vol. 46, no. 7, Jul. 1998

[12] H. J. Chao, "Next generation routers", in *Proc. the IEEE*, vol. 90, no. 9, Sep. 2002.

[13] H. Yu, S. Ruepp, and M. S. Berger, "Round-robin based multicast scheduling algorithm for input queued high-speed Ethernet switches", in *Proc. OPNETWORK2010*, Aug. 2010

[14] OPNET Modeler, available at: http://www.opnet.com/solutions/network_rd/modeler.html