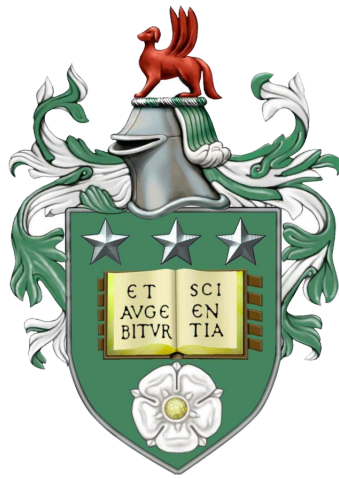


Multistage Packet-Switching Fabrics for Data Center Networks



Fadoua HASSEN

Department of Electronic and Electrical Engineering

University of Leeds

A thesis submitted for the degree of

Doctor of Philosophy

April 2017

This page is intentionally left blank.

Declaration

The candidate confirms that the work submitted is her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others. Most materials contained in the chapters of this thesis have been previously published in research articles written by the author of this work (Fadoua HASSEN), who appears as lead (first) author in all of them. The research has been supervised and guided by Dr Lotfi MHAMDI, and he appears as a co-author on these articles. All the materials included in this document is of the author's entire intellectual ownership.

The work in Chapter **3** has appeared in publication as follows:

- F. HASSEN, and L. MHAMDI, “A Clos-Network Switch Architecture Based on Partially-Buffered Crossbar Fabrics”, in *IEEE 24th Annual Symposium on High-Performance Interconnects*. Aug 2016, pp. 45–52.

The work in Chapter **4** has appeared in publication as follows:

- F. HASSEN, and L. MHAMDI, “A Scalable Multi-Stage Packet-Switch for Data Center Networks”, *IEEE Journal of Communications and Networks*, Feb 2017, vol.19, no 1, pp. 65–79.

- F. HASSEN, and L. MHAMDI, “A Multi-Stage Packet-Switch Based on NoC Fabrics for Data Center Networks”, in *IEEE Globecom Workshops*, Dec 2015, pp. 1–6.

The work in Chapter 5 has appeared in publication as follows:

- F. HASSEN, and L. MHAMDI, “Congestion-Aware Multistage Packet-Switch Architecture for Data Center Networks”, in *IEEE GLOBECOM Proceedings*. Dec 2016, pp. 1–7.

The work in Chapter 6 has appeared in publication as follows:

- F. HASSEN, and L. MHAMDI, “A scalable Packet-switch based on Output-Queued NoCs for Data Centre Networks”, in *IEEE International Conference on Communications*. May 2016, pp. 1–6.
- F. HASSEN, and L. MHAMDI, “Providing Performance Guarantees in Data Center Network Switching Fabrics”, in *IEEE 17th International Conference on High Performance Switching and Routing*. Apr 2016, pp. 155–161.

The work in Chapter 7 has appeared in publication as follows:

- F. HASSEN, and L. MHAMDI, “High-Capacity Clos-Network Switch for Data Center Networks”, in *IEEE International Conference on Communications*. 2017.
- F. HASSEN, and L. MHAMDI, “High-radix Packet-Switching Architecture for Data Center Networks”, in *IEEE International Conference on High Performance Switching and Routing*. 2017.

Dr Lotfi MHAMDI is the co-author of all the publications listed above. These publications have been written under his supervision, benefiting from his technical and editorial advice, as well as his patient guidance and valuable feedback.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

©2017 The University of Leeds and Fadoua HASSEN

The right of Fadoua HASSEN to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

To

My parents,

My brothers,

My love,

The memory of grandma

&

To

Those who said I can't

Acknowledgements

“Plus on s’approche de la fin, plus il y a de choses à dire[†]. La fin n’est qu’imaginaire, c’est une destination qu’on s’invente pour continuer à avancer, mais il arrive un moment où on se rend compte qu’on n’y parviendra jamais. Il se peut qu’on soit obligé de s’arrêter, mais ce sera uniquement parce qu’on sera à court de temps. On s’arrête, mais ça ne veut pas dire qu’on soit arrivé au bout[‡].”

— Paul Auster
— Traduction de Patrick Ferragut
Le voyage d’Anna Blume,
1987

Luckily, my journey was much pleasant than Anna Blume’s one, although I share many of her findings. I would like to thank everyone who made this voyage peaceful and joyful.

Dr. Lotfi — Thank you for your support, and for your help. You have been the advisor and friend that this journey brought to me.

Papa — You are my source of inspiration, and my flame. You are the man of all time. I would never have words to express what you are to me. “Je t’aime”.

Mama — I cherish your patience and your unconditional love. I know it has been quite long time that we did not sit for a coffee and have the long mother and daughter conversation. But it’s never too late. I inherited all of your dreams. I will promise that I will keep a smile on your face and a joy on your heart.

[†]“The close you approach the end, the more you have to say. The end is just imaginary. It is the destination that we invent to carry on moving forward. But, there comes a time when you realize that you will never reach the end. You may have to stop just because you run out of time. You stop, but that does not mean that you reached the end.”

[‡]The original text appears in *In the Country of Last Things* by Paul Auster that was first published in 1987. A translation into french was provided by Patrick Ferragut, and was published under the title *Le voyage d’Anna Blume* in 1989

Feker, Helmi and Yassine — I am indebted to you, for your support. I would never have been able to meet the challenge of completing my studies without them all. You lighten my life up, and make me proud. Big thanks to my little one, *Iyed* — who melts my heart with his mythic smile. “*Now repeat after me: ‘Paaapa, Maaaama ’ (a moment of silence while he stares at me, and here we go !)* ‘*Tatata tatata...’*. ”.

I am also grateful to *Wael* — who did never loose faith in me. You have been the trusted person on whom I rely. Thank you for being so patient, for listening to my daily complaints, and for helping me go through tough times.

I appreciate my friends who have been texting and calling me to see how life and research are going. *Imen* — thank you for some precious words you have been telling me whenever I felt short of motivation: “*J’ai confiance en toi !*”.

My colleagues — It has been a pleasure to be in the same workplace with talented people who have made the office time nice and interactive. To me, every day was a chapter of story. I met smart individuals and conversational superstars. Far away from the *I3S*, I also met other friends who made my Ph.D. life sweet.

Abstract

Recent applications have imposed stringent requirements within the Data Center Network (DCN) switches in terms of scalability, throughput and latency. In this thesis, the architectural design of the packet-switches is tackled in different ways to enable the expansion in both the number of connected endpoints and traffic volume.

A cost-effective Clos-network switch with partially buffered units is proposed and two packet scheduling algorithms are described. The first algorithm adopts many simple and distributed arbiters, while the second approach relies on a central arbiter to guarantee an ordered packet delivery.

For an improved scalability, the Clos switch is build using a Network-on-Chip (NoC) fabric instead of the common crossbar units. The Clos-UDN architecture made with Input-Queued (IQ) Uni-Directional NoC modules (UDNs) simplifies the input line cards and obviates the need for the costly Virtual Output Queues (VOQs). It also avoids the need for complex, and synchronized scheduling processes, and offers speedup, load balancing, and good path diversity.

Under skewed traffic, a reliable micro load-balancing contributes to boosting the overall network performance. Taking advantage of the NoC paradigm, a wrapped-around multistage switch with fully interconnected Central Modules (CMs) is proposed. The architecture operates with a congestion-aware routing algorithm that proactively distributes the traffic load across the switching modules, and enhances the switch performance under critical packet arrivals.

The implementation of small on-chip buffers has been made perfectly feasible using the current technology. This motivated the implementation of a large switching architecture with an Output-Queued (OQ) NoC fabric. The design merges assets of the output queuing, and NoCs to provide high throughput, and smooth latency variations. An approximate analytical model of the switch performance is also proposed.

To further exploit the potential of the NoC fabrics and their modularity features, a high capacity Clos switch with Multi-Directional NoC (MDN) modules is presented. The Clos-MDN switching architecture exhibits a more compact layout than the Clos-UDN switch. It scales better and faster in port count and traffic load.

Results achieved in this thesis demonstrate the high performance, expandability and programmability features of the proposed packet-switches which makes them promising candidates for the next-generation data center networking infrastructure.

Abbreviations

ASIC	Application-Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
BCN	Backward Congestion Notification
CCF	Critical Cell First
CIOB	Combined Input Crosspoint Buffered
CIOQ	Combined Input Output Queued
CMOS	Complementary Metal Oxide Semiconductor
CMSD	Concurrent Master Slave Dispatching
CONGA	CONGestion Aware load balancing
CPU	Central Processing Unit
CRRD	Concurrent Round Robin Dispatching
CRRD-OQ	Concurrent Round Robin Dispatching with Open Grants
DCN	Data Center Network
DEMUX	DEMUltipleXer
DISTRO	Distributed STatic ROund-Robin
DOR	Dimension Order Routing
DRAM	Dynamic Random Access Memory
DSRR	Desynchronized Static Round Robin
FCFS	First Come First Served
FCT	Flow Completion Time
FECN	Forward Explicit Congestion Notification
FIFO	First In First Out
FPGA	Field programmable Gate Array
FRD	Frame occupancy-based Random Dispatching

GALS	Globally Asynchronous Locally Synchronous
GBVOQ	Group By Virtual Output Queue
HoL	Head of Line
HW	Hardware
ICT	Information and Communications Technology
iLQF	iterative Longest Queue First
iOCF	iterative Oldest Cell First
IoT	Internet of Things
IP	Internet Protocol
iSlip	iterative SLIP
LAN	Local Area Network
MDN	MultiDirectional NoC
MIN	Multistage Interconnect Networks
MMM	Memory Memory Memory
MOMD	Maximal Oldest-cell-first Matching Dispatching
MPTCP	Multipath Transmission Control Protocol
MSM	Memory Space Memory
MUCF	Most Urgent Cell First
MUX	MULTipleXer
MWMD	Maximum Weight Matching Dispatching
NoC	Network on Chip
OCF	Oldest Cell First
OLC	Output Line Card
OM	Output Module
OP	Output Port
OQ	Output Queue
OS	Output Scheduler
PBC	Partially Buffered Crossbar
PCRRD	Pipeline-based Concurrent Round Robin Dispatching

PE	Processing Element
PIM	Parallel Iterative Matching
QoS	Quality of Service
RAM	Random Access Memory
RCA	Regional Congestion Awareness
RD	Random Dispatching
RR	Round Robin
RTL	Register Transfer Level
RTT	Round Trip Time
SDN	Software Defined Networking
SDRAM	Synchronous Dynamic Random Access Memory
SF	Scheduled Fabric
SLIP	("slip" refers to the desynchronization of grant arbiters)
SMM	Space Memory Memory
SN	Sequence Number
SRAM	Static Random Access Memory
SRR	Static Round Robin
SoC	System on Chip
S^3	Space Space Space
TDM	Time Division Multiplex
TIA	Telecommunications Industry Association
ToR	Top of Rack
TSBCS	Three-Stage Buffered Clos-network Switch
UDN	UniDirectional NoC
VC	Virtual Channel
VCMQ	Virtual Central Module Queue
VLSI	Large Scale Integrated circuits
VOMQ	Virtual Output Module Queue
VOQ	Virtual Output Queue

WUDN

Wraparound UniDirectional NoC

WWW

World Wide Web

List of symbols

AS	Accept Scheduler
BD	Buffer Depth
CA	Congestion Aware
CB	Crosspoint Buffer
CM	Central Module
CQ	Credit Queue
GS	Grant Scheduler
ILC	Input Line Card
IM	Input Module
IOM	Input Output Module
IP	Input Port
IQ	Input Queue
IS	Input Scheduler
MR	Mini-Router
NI	Network Interface
OLC	Output Line Card
OM	Output Module
OP	Output Port
OQ	Output Queue
OS	Output Scheduler
PoP	Proportion of Packets
SE	Switching Element
SP	Speedup

Contents

1	Introduction	1
1.1	Overview	1
1.2	The data center network	2
1.3	Requirements for data center networks	4
1.4	Research goals	6
1.5	List of publications	7
1.6	Thesis outline	8
1.7	Summary list	9
2	Switching and congestion-control: State-of-the-art	11
2.1	Introduction	11
2.2	When packet buffering is required	12
2.2.1	Output queuing	12
2.2.2	Input queuing	14
2.2.3	Combined input output buffering	16
2.2.4	Buffered fabrics	16
2.3	Switching architectures	17
2.3.1	Single-stage switches	18
2.3.2	Space-division switch fabric: Crossbar	22
2.4	Multistage switches	22
2.4.1	The Clos-network	25
2.4.2	In-sequence packet delivery	31
2.5	Packet scheduling	33
2.5.1	Maximum size matching	35
2.5.2	Maximum weight matching	36
2.5.3	Practical maximal size matching	36
2.6	Networks-on-Chip: Concept for IP packet switching	39
2.6.1	Packet-switching strategy	42
2.6.2	On-chip routing	45
2.6.3	Deadlocks	46
2.7	Congestion-control	47
2.7.1	Reactive congestion-control	47

2.7.2	Proactive congestion-control	48
2.7.3	Flow-control	49
2.8	Performance metrics	50
2.9	Summary list	50
2.10	Conclusions	52
3	A partially-buffered Clos packet-switching fabric	53
3.1	Introduction	53
3.1.1	Packet buffers in multistage switches	54
3.1.2	Buffers everywhere	55
3.1.3	Few internal buffers	56
3.2	Architectural design challenges	58
3.2.1	Over-provisioning in conventional MMM switches	59
3.2.2	In-order packets forwarding	60
3.3	PBClos: High-level switching architecture terminology	60
3.3.1	Distributed scheduling in the PBClos switch	62
3.3.2	Grant and accept scheduling	64
3.4	Centralized packet scheduling	67
3.4.1	Input scheduling	68
3.4.2	Output scheduling	69
3.5	Performance analysis	69
3.6	Related work	74
3.7	Summary list	75
3.8	Conclusions	76
4	A packet-switching architecture with uni-directional NoC fabric	77
4.1	Introduction	77
4.2	Clos-UDN switch architecture	78
4.2.1	The switch model	78
4.2.2	NoC based central modules	79
4.2.3	Head-of-Line blocking	82
4.2.4	Packet dispatching and scheduling in multistage switches	83
4.3	Hardware requirements	85

CONTENTS

4.3.1	Dispatching time	86
4.3.2	Hardware complexity for packet dispatching	88
4.4	Resolve the out-of-order packets delivery	89
4.5	Performance evaluation	91
4.5.1	Clos-UDN versus MSM and MMM	92
4.5.2	Further analysis of the Clos-UDN switch	95
4.5.3	Performance of the two-stage Clos-UDN switch	97
4.6	Related work	102
4.7	Summary list	104
4.8	Conclusions	106
5	A Congestion-Aware Clos-UDN switch	108
5.1	Introduction	108
5.2	Congestion-aware load-balancing in DCNs	109
5.2.1	Micro load-balancing	110
5.3	Description of the switch architecture	111
5.3.1	Cross-module interconnection	112
5.4	Congestion-aware routing	113
5.4.1	Congestion evaluation	114
5.4.2	The repellent routing	116
5.5	Implementation issues	119
5.5.1	The architecture of the on-chip routers	119
5.5.2	Re-ordering packets	120
5.6	Simulation results	120
5.7	Related work	124
5.8	Summary list	125
5.9	Conclusions	126
6	A multistage switching fabric with Output-Queued NoC modules	128
6.1	Introduction	128
6.2	Clos-UDN switch with output-queued mini-routers	129
6.2.1	Nomenclature of the switching architecture	129
6.2.2	Packet switching mode in CMs	131

6.2.3	Routing in the OQ-UDN modules	132
6.2.4	Implementation feasibility of the switch	132
6.3	Analytical modelling of the switch performance	134
6.3.1	Characterization of the throughput of the Clos-UDN switch	139
6.3.2	Average end-to-end delay	140
6.3.3	Blocking probability in the switch	142
6.4	Performance evaluation	143
6.4.1	Uniform packet arrivals	143
6.4.2	Unbalanced traffic	147
6.4.3	Scalability of the switch	148
6.4.4	Accuracy of the analytical model	150
6.5	Summary list	152
6.6	Conclusions	153
7	Clos-MDN: A multistage packet-switch with multi-directional NoC fabric	154
7.1	Introduction	154
7.2	Motivation	156
7.3	High-level architecture	158
7.3.1	Network topology and packet buffers	158
7.3.2	Packet routing in the MDNs	160
7.3.3	Proactive congestion control	161
7.4	Performance Analysis	163
7.4.1	Uniform traffic	163
7.4.2	Non-uniform traffic	164
7.5	Summary list	170
7.6	Conclusions	171
8	Conclusions	172
8.1	Research contributions	175
8.2	Concluding remarks	177
8.3	Prospective work	178
8.3.1	Future directions	178
8.3.2	Open problems	180

CONTENTS

A	Traffic types and performance indices	181
A.1	Traffic types	181
A.1.1	Uniform traffic	181
A.1.2	Non-uniform traffic	182
A.2	Performance indices	183
B	Upper bound on the blocking probability	185
C	Analytical modelling of the Clos-MDN switch's performance	189
C.1	Structure of an MDN module	189
C.2	Assumptions	189
C.3	Outline of the model	191
C.3.1	End-to-end latency	191
C.3.2	A single queue in the tandem	193
C.3.3	Exogenous arrival rates	194
	References	220

List of Figures

1.1	DC traffic growth [4].	1
1.2	Cost structure in a DC [5].	1
1.3	Conventional multi-tier tree DCN topology.	3
2.1	The throughput of an input queued switch saturates at around 58% if simply FIFO queues are used at the input side. Using Virtual Output Queues (VOQs), the Parallel Iterative Matching (PIM) [20] improves the performance a little, and results in about 63% of the maximum achievable throughput. An Input-Queued (IQ) switch with VOQs and a non-iterative iSlip algorithm [21] achieves full throughput, but results in much higher packet delay than the Output-Queued (OQ) switch.	13
2.2	An $N \times N$ switch with VOQs. The HoL blocking problem can be alleviated using VOQs. There are N separate VOQs per input port that operate according to the FIFO discipline. At the end of each cycle, the scheduling algorithm decides which VOQ to serve per input, and the HoL packet is removed from the input queue to be forwarded to the switching fabric, and then to the output line card.	15
2.3	An example of a shared memory fabric [42].	19
2.4	An example of shared bus fabric [48].	20
2.5	An example of output buffered switch fabrics with N^2 disjoint paths [48].	21
2.6	Crossbar switching fabric.	23
2.7	A three-stage Clos interconnect.	25
2.8	Space-Space-Space Clos-network switching fabric.	27
2.9	Contention at IPs.	29
2.10	Contention for CMs.	29
2.11	Contention for the OPs.	29

LIST OF FIGURES

2.12	An example of a Request-Grant-Accept matching used in the PIM algorithm [80]. Unmatched inputs send requests to outputs for which they have a packet. Outputs grant one of the received requests, and ties are broken randomly. Input ports also select one of the returned grants in a random fashion. Note that at the end of the first iteration, input 2 remains unmatched, although it does not conflict with any other connection. It would be matched at the second iteration [19].	37
2.13	Examples of regular Networks-on-Chip topologies.	41
2.14	An example of the store-and-forward switching.	43
2.15	An example of the virtual cut-through switching.	44
2.16	An example of the wormhole switching.	44
2.17	The flow control ensures that the switch is working fine and that any buffers saturation is avoided [22]. The control signal can be back forwarded from the internal buffers of the switching fabric (1bit/queue)or output buffers that are associated to the switch egresses (N bits/priority).	49
3.1	$N \times N$ buffered crossbar switch with N^2 crosspoint buffers. . .	57
3.2	$N \times N$ Partially buffered crossbar switch with $B \ll N$ physically separate internal buffers. Internal buffers are logically shared among all input queues. Distributed arbiters work in parallel to manage the packets transfer to and out of the PBC fabric. . . .	58
3.3	$N \times N$ three-stage PBClos switch architecture.	61
3.4	PBClos switch with distributed scheduling. Grants are forwarded from CMs to active VOQs. A VOQ that receives more than one grant from a CM (<i>i.e.</i> , The CM has room to house a packet to the corresponding OP), accepts one grant in a RR fashion. . . .	62
3.5	Scheduling in the central stage PBC switches.	63

LIST OF FIGURES

3.6	Pipelined scheduling in the multistage PBClos switch using distributed schedulers. Internal buffers are managed using a credit-based flow-control. The req-grant Round Trip Time (RTT) starts from the time requests are sent to CMs, until grants are received at input ports. Buffers reservation RTT spans the instant credits are reserved in the CM switching fabric, until they are released when packets finally exit the switch.	64
3.7	An example of batch scheduling in the PBClos.	66
3.8	Centralized control in PBClos packet scheduling.	68
3.9	Delay performance for a 256×256 switch under <i>Bernoulli</i> uniform traffic.	71
3.10	Delay performance for a 256×256 switch under <i>Bursty</i> uniform traffic.	72
3.11	Delay performance of a 64×64 switch under <i>hot-spot</i> traffic. . .	73
3.12	Throughput stability of 256×256 switch under <i>non uniform</i> traffic, $B = k/2$	74
4.1	$N \times N$ three-stage Clos-UDN packet-switch architecture with dynamic dispatching scheme.	79
4.2	The UDN crossbar switch.	80
4.3	Pipelined working of Clos-UDN dispatching and packets forwarding through UDN modules.	83
4.4	Delay performance for a 256×256 switch under <i>Bernoulli</i> uniform traffic.	92
4.5	Performance of Clos-UDN and MSM switches, under <i>uniform</i> traffic.	93
4.6	Performance of a 64-ports switch under <i>unbalanced</i> traffic. . . .	94
4.7	Switch performance, under <i>Bernoulli</i> traffic.	96
4.8	Impact of the speedup SP, and mesh depth M, on the Clos-UDN switch latency, <i>Bursty</i> traffic.	97
4.9	Throughput stability of the two-stage and three-stages Clos-UDN under <i>Unbalanced</i> traffic.	97

LIST OF FIGURES

4.10	An example of a 9×9 Clos-UDN switch with static configuration of the IMs/CMs interconnections.	98
4.11	Delay performance of a 64-ports two-stage Clos-UDN switch under uniform traffic.	99
4.12	Delay performance of a 64-ports switch under <i>Hot – Spot</i> traffic.	100
4.13	Proportion of Packets moving west-east in the middle-stage UDNs.	101
5.1	Design space of the load-balancing techniques in DCNs [165].	110
5.2	$N \times N$ Three-stage Clos switch architecture with cross-connected CMs.	112
5.3	Design space for NoC routing policies with respect to congestion avoidance.	115
5.4	Example of the Repellent and “ <i>Modulo XY</i> ” routing algorithms.	117
5.5	High-level diagram of an adaptive on-chip router.	118
5.6	Delay performance in the Congestion-Aware Clos-UDN switch under <i>Bernoulli</i> uniform traffic.	121
5.7	Delay performance under <i>bursty</i> and <i>hot – spot</i> traffic.	122
5.8	Delay performance of 64×64 switches under <i>diagonal</i> traffic.	123
5.9	Throughput stability of different switches under <i>non – uniform</i> traffic.	124
6.1	An $N \times N$ three-stage OQ Clos-UDN packet-switch architecture.	130
6.2	Routing process in an a mini-router of the OQ-UDN central module.	133
6.3	A block diagram of an Output-Queued mini-router.	135
6.4	The state transition diagram for a single M/M/1/B queue.	136
6.5	Types of mini-routers in an OQ-UDN switch based on their degrees.	139
6.6	Packet delay in a tandem queue.	141
6.7	Delay performance of single-stage and multistage 64×64 switch, under <i>Bernoulli</i> uniform traffic, $B = 3$	144
6.8	Delay performance of single-stage and multistage 256×256 switch, under <i>Bernoulli</i> uniform traffic, $B = 3$	144
6.9	Delay performance of different 256×256 switches, under <i>Bernoulli</i> uniform traffic, $B = 3$	145

LIST OF FIGURES

6.10 Delay performance of single-stage OQ-UDN 64×64 switch, under uniform <i>Bursty</i> traffic.	145
6.11 The average latency of MSM, MMM,IQ Clos-UDN and OQ Clos-UDN, for 256×256 switch size, under uniform <i>Bursty</i> traffic, $B = 3$	146
6.12 Delay performance of 64×64 MSM, MMM, IQ Clos-UDN and OQ Clos-UDN, under <i>Unbalanced</i> traffic, $B = 3$, $\omega = 0.5$	146
6.13 Variation of the transfer and blocking probability in the central modules of a 64×64 switch.	148
6.14 Throughput stability of 64-ports switches.	149
6.15 Impact of the switch size on the packet delay under <i>hot-spot</i> traffic.	149
6.16 Average throughput for various switch valency, under <i>Bernoulli</i> uniform traffic.	150
6.17 The average end-to-end delay in a 256×256 switch under <i>Bernoulli</i> uniform traffic.	150
6.18 Variation of the blocking probability in 64×64 switch under <i>Bernoulli</i> uniform traffic.	151
7.1 A high-level diagram of the central-stage MDN module.	155
7.2 Requirements of the Clos-UDN/Clos-MDN switches.	157
7.3 Generic layout of the Clos-MDN switching architecture.	159
7.4 An example of an 8×8 Clos-MDN switch.	159
7.5 Symmetrical/asymmetrical buffers distribution in the MDN mini-routers.	160
7.6 Performance for 256-ports Clos-UDN/MDN Switches.	165
7.7 Performance for 256-ports switches under Bernoulli hot-spot traffic, $\omega = 0.5$	165
7.8 Delay performance for 256-ports Clos-MDN Switches under non-uniform bursty traffic.	166
7.9 Performance of 256-ports MSM, MMM and Clos-MDN Switches.	167
7.10 Throughput performance of 256-ports MSM, MMM and Clos-MDN Switches.	168
7.11 Impact of switch size on performance of Clos-MDN Switch, $SP = 3$	168

LIST OF FIGURES

7.12 Impact of switch size on performance of Clos-MDN Switch, $SP = 3$, <i>Bursty</i> non-uniform.	169
8.1 Contributions of the thesis.	176
C.1 The MDN can be viewed as the superposition of two UDNs with opposite horizontal traffic flow directions. It implements VCs to convey packets to their destinations based on their addresses.	190
C.2 A tandem of queues.	191
C.3 Equivalent model of the tandem of queues.	193

List of Tables

2.1	Comparison between the different switching architectures [65, 66].	28
2.2	Examples of scheduling algorithms for different switches.	31
2.3	Shared bus versus Network-on-chip [114].	40
2.4	Asymptotic cost function [117].	40
2.5	Points of similarities and differences between NoCs and computer networks.	41
3.1	Comparison between the different switching architectures.	70
4.1	The terminologies for the Clos-UDN switch.	80
4.2	Compare the HW requirements of MSM, MMM and Clos-UDN switches.	86
4.3	HW requirements: Dynamic vs static dispatching in Clos-UDN packet-switch.	91
C.1	List of symbols used for the model.	192

List of Algorithms

1	“Modulo XY ” routing	82
2	The repellent routing	118
3	Inter-CM interleaved connections	162
4	Iterative exogenous arrival rates computation	198

1

Introduction

1.1 Overview

Since the appearance of computers, the need for speedy and efficient computing remarkably increased. Many of the sensitive and dynamic fields – especially scientific fields – such as high-energy physics, biology, medical imagery and the world-wide communications, rely on sharing the universal computer reservoir across the international grids to meet the huge requirements of massive amounts data processing. The last computing enhancement to notice, was the cloud computing which depends on the Internet to provide access services on virtual resources housed in Data Centers (DCs). A data center is the warehouse that regroups computing facilities such as servers, switches/routers and firewalls, as well as other equipment, like instruments of air cooling and fire extinguishers, etc. Data centers are phenomenally growing in size and complexity [1–3], and the emergence of demanding applications such as the Internet of Things (IoT) and big data significantly increased the traffic within and in between DCs [4]. As shown in Figure 1.1, the annual Internet Protocol (IP) traffic in a DC is

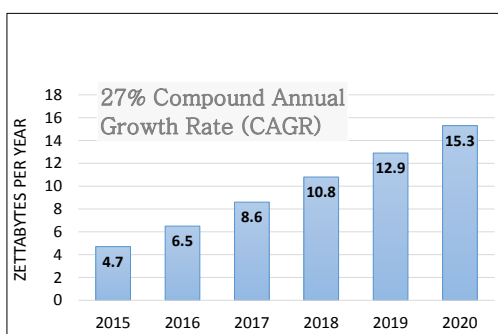


Figure 1.1: DC traffic growth [4].

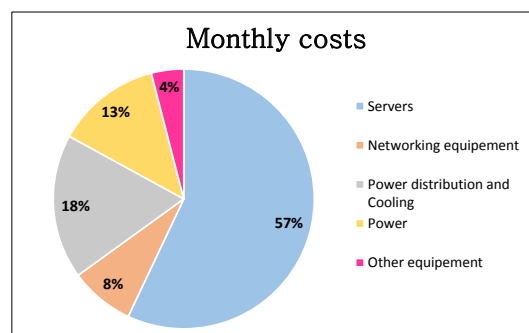


Figure 1.2: Cost structure in a DC [5].

expected to reach 15.3 Zettabytes by the end of year 2020 – which is more than

1. INTRODUCTION

three-fold increase from year 2015. Information and Communication Technology (ICT) organizations like IBM, Microsoft, Amazon, Facebook, and Google, are continuously deploying hundreds of servers, and extending the computational operation of their DCs. However, the performance of a data center do not rely only on its computational capacity. It is inextricably dependent on the capability of the interconnection network infrastructure which is still a major bottleneck. The monthly cost of the networking equipment in a data center is considerable¹. It is estimated to be 8%, as shown in Figure 1.2. Largely, the capital cost of the DC networking is concentrated in switches, routers, and load balancers [6, 7]. The switches and routers are the first traffic coordinators in the DCN. They are also the first elements to get affected by the growth of the network connections and the increase of the traffic volume. Small-radix switches/routers deployed in the current DCNs, exhibit limited scalability features. They constraint the throughput and penalize the scalability of the whole data center [8]. Thus, innovation in the field of the switching architectures design is paramount to enable the scalable growth in – both – the number of connected endpoints and exchanged traffic volume. Optical switching solutions combined with advances in the Complementary Metal Oxide Semiconductor (CMOS) technology, allowed the implementation of high-radix switches [9]. Other modular approaches such as the multistage, also helped design large and cost-effective electronic switches [10–13].

1.2 The data center network

The DC interconnection network has to dynamically support large amounts of workloads flowing in between the servers. Traditional DCNs are built in hierarchical multi-tier tree fashion where servers are lodged in a number of racks as shown in Figure 1.3. These racks are grouped into clusters. They exchange traffic with each other via Top-of-Rack (ToR) switches. Each ToR switch is usually connected to sets of aggregation switches within the same cluster for better redundancy [8].

¹Not to be compared to the cost of servers [5].

1.2 The data center network

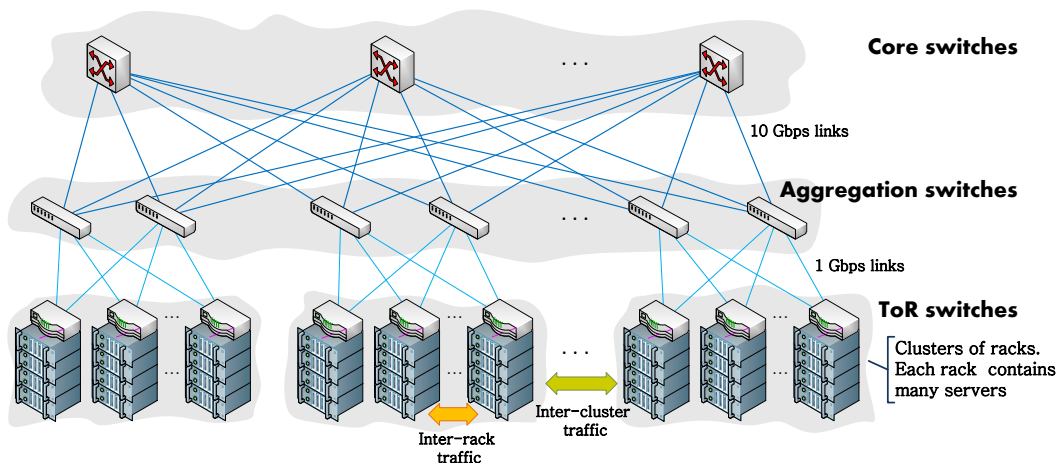


Figure 1.3: Conventional multi-tier tree DCN topology.

Rebuilding redundancy into the DCN switches and links ameliorates the path diversity, and prevents the network from decaying in case a component breaks down. The aggregation-level switches – in turn – are linked to each other through core switches. The multi-tier architecture has several advantages such as being easily expandable¹ and fault-tolerant². It – also – exhibits good path diversity which helps avoid the sudden collapse of the network performance in the event of failure [14]. Yet, multi-tier tree DCNs suffer the handicap of oversubscription which threatens its performance, and leads to congestion. Ideally, the network should provide a full bisection³ bandwidth between the end hosts [15]. This can be done by scaling in bandwidth and port density of the switches and routers. However, in the presence of conventional electronic switches, such an action would be prohibitively expensive due to the super linear

¹The multi-tiered DCN architecture can be scaled to provide larger connectivity in a cost effective manner [14].

²Hardware failure is common in huge communication networks. In DCNs, equipment can fail at any time which makes the network low utilizable and poorly reliable. A straightforward solution to this problem consists on ensuring the availability of other equipment to save part or the whole of the network from crashing down. This is known as the *availability* of the DCNs.

³The bisection bandwidth refers to the available bandwidth between any two bisected sections. It is said to be full, when it equals the aggregation bandwidth of the servers in one of the sections.

1. INTRODUCTION

scaling costs. The over-subscription ratio is defined as the ratio of the worst-case aggregate bandwidth achievable among the end hosts to the total bisection bandwidth of a particular network [1]. Often, DCNs call for oversubscribed bandwidth in the access tier and – to a lesser degree – in the aggregation nodes. An over-subscription of 1:1 means that the end hosts can communicate with each other at the full bandwidth. Respectively, an oversubscription ratio of 5:1 indicates that only 20% of the available inter hosts bandwidth is open to handle some communication patterns. The value of the oversubscription ratio depends on the communication requirements of the applications. In practice, traditional data centers tend to oversubscribe their connectivity with a factor that ranges from 4 to 10 [16]. It is interesting that some networks can operate well when oversubscribed. Still, the risk of congestion is considerable. Whenever hosts on a given access switch need to communicate at high speed with other hosts related to another switch, the uplink connection (between the access layer and the aggregation layer) is much likely to become a congestion point.

Data center networks have evolved into many forms to meet the bandwidth and latency requirements, and to cope with the limitations of the primitive network topologies. In addition to agility, cost and manageability, much research has gone into developing scalable DCN architectures with focus on the switching substrate. Being the first traffic coordinators in the DCN, switches and routers are closely committed to the overall network performance. They have solicited interest to build scalable and high-performance modules capable of facing the demand of the next-generation data centers.

1.3 Requirements for data center networks

In addition to virtualization¹ and security, modern data centers require high cross-section bandwidth, low latency and simple management that can scale to thousands of servers at low cost. Generally speaking, a DC network should be scalable, expandable, manageable, and cost-effective. It must also provide high

¹The term virtualization refers to the act of decoupling the logical computational requirement from the actual physical infrastructure. The technique allows for an efficient sharing of the physical resources.

1.3 Requirements for data center networks

throughput performance, and guarantee low delays for the different running applications. The DC requirements can be briefly described as follows:

- **Scalability:** Current DCs house few hundreds of thousands of servers while most deployed switches valency can go to – only – hundreds of ports. The architectural design of the data center network is critical in many ways. Most importantly, choosing a good architecture is the first step to prevent the network from becoming a limiting factor of the number of servers.
- **Expandability:** It is important that the network does not prevent the data center from scaling in size. It should be possible to add new servers and switches/routers to the network substrate with little alteration.
- **Manageability:** The large number of servers in the DCN make it impractical and excessively costly to manage all of the interconnected devices manually. Therefore, it is ideal to use “plug-and-play” network devices that immediately join the existing network without much human intervention.
- **Cost-efficiency:** The cost metric is a primordial decision factor in the DC deployment. The capital investment of a data center is dominated by the DCN devices cost (switches/routers, links, load-balancers, power suppliers, etc.) as well as operation costs (power consumption and devices management). Data centers also tend to make profit at first place making the cost a primary concern.
- **Overall performance:** In addition to the one-to-one traffic, many critical applications, frameworks and services, generate acute traffic patterns which need be supported in an effective way. Since all applications and services run simultaneously in a data center, the switching network of the data center should never be the limiting factor in the transmission rate between end-hosts. Consequently, users should receive guaranteed bandwidth regardless of the traffic conditions.

1.4 Research goals

Today, no industry is exempt from being overwhelmed by data. Internet, social networking, services on the Cloud, etc., massively contribute to the amplification of the data traffic. However, the on-operation DCNs build with common switches/routers show restricted ability to scale with the proliferating traffic loads. The switching fabric is an important layer in a data center network. It accommodates the fundamental hardware that effectively processes the data traffic upon its arrival to the network. Switches and the routing units largely determine the overall cost and performance of the DCN. Therefore, designing switches that respond to the requirements of DCs have attracted much attention. To face the increasing bandwidth demand while restricting the power consumption in the data center, the switches and routers must be capable of providing high throughput, low latency as well as a small energy footprint. Traditional Ethernet switches commonly deployed in DCNs, cannot satisfy the soaring requirements at a large scale. However, some enterprises choose to go for common switches to match up with the existing hardware.

The main goal in this thesis is to design scalable packet-switching architectures that better respond to the next-generation networking demand. The following reasons – and several others – explain why DCNs need incorporate scalable switches/routers.

1. Data centers are the crux of storage, networking and communications between a growing number of interconnected devices. The rising importance of data traffic flowing from ubiquitously networked users contributed to the growth of the DCNs and increased the need for a better network monitoring.
2. Enhancing the DCN response to traffic fluctuation has a direct effect on the performance as perceived by the end-users who is primarily concerned about the performance of their own applications.
3. It is a good idea to make the monitoring part of the data center's architecture. Scalable switching units used along with over-subscription

tools, are often able to handle the traffic increase, and to alleviate the need to buy new hardware.

1.5 List of publications

The original contributions in this thesis are supported by the following publications:

Journal papers

- J1- F. HASSEN, and L. MHAMDI, “A Scalable Multi-Stage Packet-Switch for Data Center Networks”, *IEEE Journal of Communications and Networks*, Feb 2017, vol.19, no 1, pp. 65–79.

Conference papers

- C1- F. HASSEN, and L. MHAMDI, “A Clos-Network Switch Architecture Based on Partially-Buffered Crossbar Fabrics”, in *IEEE 24th Annual Symposium on High-Performance Interconnects*. Aug 2016, pp. 45–52.
- C2- F. HASSEN, and L. MHAMDI, “A Multi-Stage Packet-Switch Based on NoC Fabrics for Data Center Networks”, in *IEEE Globecom Workshops*, Dec 2015, pp. 1–6.
- C3- F. HASSEN, and L. MHAMDI, “A scalable Packet-switch based on Output-Queued NoCs for Data Centre Networks”, in *IEEE International Conference on Communications*. May 2016, pp. 1–6.
- C4- F. HASSEN, and L. MHAMDI, “Providing Performance Guarantees in Data Center Network Switching Fabrics”, in *IEEE 17th International Conference on High Performance Switching and Routing*. Apr 2016, pp. 155–161.
- C5- F. HASSEN, and L. MHAMDI, “Congestion-Aware Multistage Packet-Switch Architecture for Data Center Networks”, in *IEEE GLOBECOM Proceedings*. Dec 2016, pp. 1–7.
- C6- F. HASSEN, and L. MHAMDI, “High-Capacity Clos-Network Switch for Data Center Networks”, in *IEEE International Conference on Communications*. 2017.
- C7- F. HASSEN, and L. MHAMDI, “High-radix Packet-Switching Architecture for Data Center Networks”, in *IEEE International Conference on High Performance Switching and Routing*. 2017.

Specifically, Chapter **3** develops the ideas reported in the conference paper C1. The work in Chapter **4** appeared in papers C2 and J1. The work presented in the conference paper C5 forms the core of the Chapter **5**. Chapter **6** develops the framework proposed in papers C3 and C4, as well as J2. Finally, Chapter **7** is related to the piece of work presented in papers C6 and C7.

1.6 Thesis outline

In this thesis the high-performance multistage packet-switching architectures that are suitable to meet the networking infrastructure requirements of DCNs are studied. In particular, many related issues such as the packet buffering, packet scheduling algorithms and the control mechanisms are addressed. In Chapter **2**, the state-of-the-art multistage switching fabric design, and related work are reviewed. In Chapter **3**, a buffered three-stage Clos switch is highlighted. The concept is borrowed from the single-stage partially buffered switch to redesign the central stage modules. Making trade-off between the design cost, practicality and performance of the switch, different scheduling algorithms are proposed to manage packets transfer with and without in-order forwarding guarantee.

In Chapter **4**, a novel way to build a non-blocking multistage Clos switch using the Networks-on-Chip paradigm is presented. The approach combines a Clos macro design, and a NoC micro design. Unlike common crossbar-based switches, the Clos-UDN switch is a nested network with much better path diversity, pipelined/parallel packets scheduling/transfer, and better scalability. The performance of the switch is compared to the classical Memory-Space-Memory (MSM) and Memory-Memory-Memory (MMM) switches, to attest of the pros of a NoC-based design. In Chapter **5**, a multistage switch with cross-connected NoC modules is proposed, along with a sophisticated congestion-aware routing algorithm. The CA Clos-UDN switch tends to better balance uneven loads between the central-stage modules. It contributes to lower the congestion level, and to boost the switch throughput.

Next, an output-queued Clos-UDN switch is described. The design is provided in Chapter **6** and it exhibits an OQ NoC switching fabric. Details of this multistage architecture, specifications of the routing algorithm, as well as

a simplistic mathematical model for the switch performance are also given in Chapter 6. Chapter 7 is concerned with a Clos-MDN switch made of multi-directional NoC modules. The switch has a more compact design¹, and scales faster than the Clos-UDN and the CA Clos-UDN switches. It makes full use of the interesting features of the Networks-on-Chip paradigm (speedup, virtual channels, etc.). Ultimately, in Chapter 8 the main conclusions of the thesis are presented. Furthermore, in this chapter a few possible directions of future research aligned with the work presented are outlined.

1.7 Summary list

- 1- Data centers contain hundreds to thousands of servers in an economy of scale, that need to be effectively interconnected.
- 2- The architectural design of the data center affects its overall performance. Thus, it is no more a choice to opt for scalable, low-cost, and robust DCNs.
- 3- It is all important that the DCN do not prevent the data center from scaling in size. In other words, it should be possible to increase the number of servers, switches, and routers in the network with little alteration.
- 4- The networking infrastructure in the DC is the foundation upon which are build all services of the data center. It allows to successfully handle the data traffic, and to serve the needs of the next-generation DC.
- 5- The goal of this thesis is to design multistage packet switching architectures that — (1) provide high-performance, (2) are easily scalable, (3) meet the challenging requirements of DCNs.
- 6- This thesis probes many aspects, to come up with efficient packet-switching fabrics that simultaneously meet more than one criterion among the following: (1) Cost-effectiveness, (2) scalability, and (3) high-performance.
- 7- A buffered three-stage Clos-network switch is designed in a way to use a restricted number of internal memories. Accordingly, two different packet scheduling solutions are proposed: (1) Distributed, and (2) centralized. The first approach is simple. It

¹As compared to previous proposals discussed within this thesis.

1. INTRODUCTION

relies on many small arbiters dispersed across the network to manage the packet transfer. The second heavily depends on a central controller to ensure an in-order packet delivery.


- 8- To build in better scalable and expandable switching fabrics, the middle-stage switching elements of the Clos-network is radically altered. Instead of – passive – internally buffered crossbars, Network-on-Chip fabric modules are plugged.
- 9- The NoC-based design overcomes many shortcomings of the conventional multistage packet-switches. More importantly, it contributes towards lowering the hardware complexity of the input line cards, and simplifying the packet scheduling process.

2

Switching and congestion-control: State-of-the-art

2.1 Introduction

The Internet amalgamates thousands of commercial and service provider networks. It established the worldwide communication medium for voice, video and data. At some point, it pushed forward the concept of computing, that widely flourished since a mass of people have had access to the global World-Wide-Web (WWW) system. Today, the Internet has mutated into other forms of networks of larger scale and more enabling technologies. Yet, all of these forms rely on packet-switched networks, and use statistical multiplexing principles to share resources among users. In a packet-switched network, end hosts, routers, and switches communicate using special protocols. These protocols set in order the IP packets transmission across the network, from a source node to a destination node. Today, the emergence of many network-related applications with stringent Quality of Service (QoS) requirements impose design challenges on switches/routers for equipment vendors and service providers. In this thesis, the packet-switching fabric architectures that makes the core of a switch/router are reviewed.

 **Packet, cell** The state-of-the-art semiconductor technologies help implement packet-switches with transmission rates up to few Gb/s using the current Very Large Scale Integrated circuits (VLSI) technology. To reduce the implementation complexity, packet-switching is usually tailored towards fixed-length packets at the hardware level. In this thesis, the term packet is considered for a fixed-size data unit. The size of a packet ranges from 32 to 256 bytes, and is generally considered to be 53-bytes – the same size as a cell used in the Asynchronous Transfer Mode (ATM) networks.

2.2 When packet buffering is required

In packet-switching buffers might be needed to temporarily hold packets for various reasons that basically depend on the Hardware (HW) design and scheduling choices. Data processing at the heart of the switch/router relies on one or many arbiters that monitor the shared resources among contending input and output ports. It may happen that the arbiter is busy serving earlier packets while current arrivals request a shared medium access. As a result, packets can be effectively buffered, using an adequate scheme, until transmitted rather than being dropped at any time a traffic oversubscription or a congestion occurs. Obviously, a switch/router is meant to be integrated into a network wherein data flows have different transmission rates which most commonly leads to transient network congestion. Therefore adding buffers to the switch/routers, comes to compensate for speed disparity, and to handle traffic bursts. In general, buffers are required whenever one or more of the following scenarios are valid:

- Speed mismatch between input and output ports
- Multiple ingress ports trying to send packets to a single egress port
- Half-duplex collision at an output port

2.2.1 Output queuing

Many of the commercial routers use the output queuing with shared central memory [17]. Packets are immediately transferred through the fabric to the corresponding output queue. The approach of output queuing tends to resolve output contention in which case, if more than just one packet arrive to the same output at a given time-slot¹, and that the switch fabric is capable of transferring up to “ X ” packets; all “ X ” packets are transferred before the next time-slot starts [18]. For such an action a set of requirements needs be in

¹A time-slot – also called packet-time or cell-time – is the back-to-back time period that a packet takes to cross the switch. To make the switch run at the same speed as the external lines, the time-slot duration is set to be equal to the time interval between the arrivals of two consecutive packets to the switch input ports.

2.2 When packet buffering is required

place. Most importantly the switch must run – at least – “ X ” times faster than the input line cards so that none of the packets get lost [19]. The output queuing is known to maximize the throughput of the switch [18] (theoretically, it achieves full throughput, as shown in Figure 2.1). Therefore unless an output is oversubscribed, the switch can support the traffic and maintain queues occupancies bounded [17]. Unlike other buffering strategies, storing packets in output buffers let all packets experience a fixed delay through the switch. Therefore it is possible to control packets’ delay by carefully scheduling them,

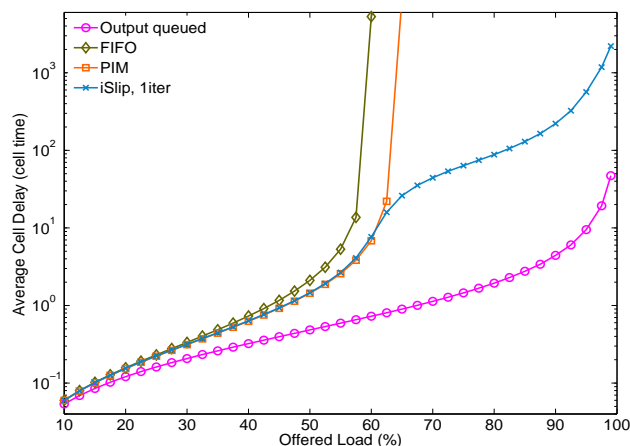


Figure 2.1: *The throughput of an input queued switch saturates at around 58% if simply FIFO queues are used at the input side. Using Virtual Output Queues (VOQs), the Parallel Iterative Matching (PIM) [20] improves the performance a little, and results in about 63% of the maximum achievable throughput. An Input-Queued (IQ) switch with VOQs and a non-iterative iSlip algorithm [21] achieves full throughput, but results in much higher packet delay than the Output-Queued (OQ) switch.*


and hence offer a better QoS. The major constraint for OQ crossbar switches, is the memory speed. Actually, OQ switches are often implemented in a shared-memory architecture. The choice is made to increase the buffer efficiency¹. However, the aggregate shared-memory bandwidth is always proportional to – both – the line rate and the number of switch ports [22]. Unfortunately, advances in the field of memory are far from running at the same pace at which

¹The tendency is to opt for a shared-memory implementation rather than dedicated output queues.

networks are evolving. This makes an OQ switch unscalable and unaffordable to build high-performance and high valency switches [17].

2.2.2 Input queuing

The input queuing is used to solve the packet contention problem at the input side. Each input has a dedicated buffer where it stores arriving cells, and an arbitration logic is used to serve the buffered data packets. The switch fabric runs at the same speed as the I/O ports, and only one packet per input port, is eligible for transmission across the switch [18]. This makes the input-buffered architectures appealing. When input buffers are used, the mean waiting time is greater than for output queuing because a cell whose output is idle might be waiting for a cell ahead of it to be transmitted, while its output is busy receiving another cell from another input. This problem is referred to as the Head of Line (HoL) blocking [19].

 **Head-of-Line Blocking** is a general phenomenon that takes place whenever a set of circumstances is valid. Consider a single-stage $N \times N$ crossbar switch, where packets arriving at a given input port are stored in the same input FIFO. Given the slotted nature of time, only one packet comes to each input among the switch input ports, and only one packet can exit an input queue to the fabric. A HoL cell at an input queue i destined to output port j cannot be transmitted unless the corresponding output line card is free. In the counter case, the packet remains at the input queue i , and blocks the bunch of cells queued behind it – although their output ports might be free for use. The HoL blocking has bad effects on the switch performance [21]. For instance, the maximum achievable throughput of an IQ switch with FIFO input queues falls to 58.6% under certain conditions¹ [18], as depicted in Figure 2.1. Several techniques have been proposed to avoid the HoL blocking in input-queued switch types. Placing the packet buffers at the output side of the switch, radically resolves the issue of HoL blocking. It is also possible to mitigate the problem by modifying the scheduling process [24, 25] or the hardware [26] or both [21, 27]. The virtual output queuing – illustrated in Figure 2.2 – where each input port maintains a separate FIFO per output port, entirely removes HoL blocking [19, 28]. Moreover, using an adequate scheduling, the switch can offer full throughput under different traffic types.

¹These conditions are the following [23]: (1) Independent and identically distributed (i.i.d.) packet arrivals happen at each input, (2) independent arrival processes at each input and inter-inputs, (3) the port count N is large, (4) the arrival rate is fixed, (5) destinations are uniformly distributed over all outputs, and (6) packets are of fixed-size.

2.2 When packet buffering is required

An input-queued switch with *Bernoulli i.i.d* arrivals and uniformly distributed destination probabilities can achieve a maximum throughput of just 58% when the number of I/O is large ($N \rightarrow \infty$) [18] (see Figure 2.1). At each time-slot, the central scheduler¹ of the IQ switch looks for packets at the head of the FIFO queues. If they request different outputs, then the scheduler picks the right match to make packets cross the switch. If more than only one packet is addressed to the same output, then the central scheduler chooses one packet to be transmitted during that time-slot. Other packets should wait for next time-slots to traverse the switch fabric. Theoretically, the input buffer saturates at a value of load $\rho < 1$. The critical ρ value is equal to $(2 - \sqrt{2} = 0.586)$ when N is large and a random selection policy is used [18]. Many architectural and scheduling solutions have been proposed to resolve the problem of HoL, such as the input queuing with a window[29], input smoothing [18], Combined Input Output Buffering (CIOB) and the virtual output queuing [23, 29], etc.

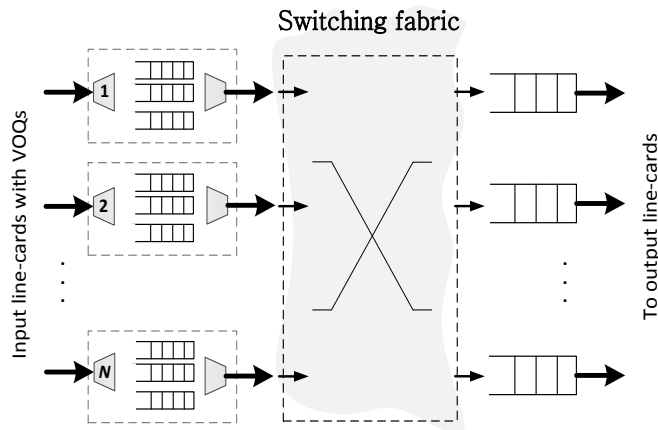


Figure 2.2: An $N \times N$ switch with VOQs. The HoL blocking problem can be alleviated using VOQs. There are N separate VOQs per input port that operate according to the FIFO discipline. At the end of each cycle, the scheduling algorithm decides which VOQ to serve per input, and the HoL packet is removed from the input queue to be forwarded to the switching fabric, and then to the output line card.

¹Assuming that the centralized scheduler inspects the state of the input queues and selects a conflict-free match between the set of input and output ports [19].

2.2.3 Combined input output buffering

The input/output queuing approach consists on introducing buffers to both inputs and outputs of the switch. Commonly FIFOs or VOQ are coupled with a bufferless crossbar or a shared memory [30]. Hence, the contention resolution is distributed over the inputs and the outputs [31], and a centralized arbiter is responsible for managing the connectivity between the I/O devices via the crossbar fabric. The CIOB needs to operate at limited speed unlike output buffered switches that need run N times faster than the external line rate. A factor of speedup S where $1 < S < N$ can be sufficient. The switch can move up to S packets from each input and transfer at most S packets to an output. It has been shown that a CIOB switch can reach up to 99% throughput under independent, uniform packet arrivals if a speedup of 4¹ is used [32]. The combined input output buffered switches need scheduling algorithms to select a matching between I/O ports [17]. To this end, many algorithms have been proposed. One can cite: Critical Cell First (CCF) [17], Group By Virtual Output Queue (GBVOQ) [17], Most Urgent Cell First (MUCF) [32], etc.

2.2.4 Buffered fabrics

The output queued switching architecture is known to have the best performance amongst all queuing approaches. It offers the best packet latency control, and hence provides good QoS management. However, output queued switches – OQ – cannot fit for high rate needs. First, they need a high internal speedup. Besides, the memory evolution in terms of speed access time and on-chip size is still limited compared to the abrupt evolution of networks. IQ switches do not need internal speedup. The crossbar works at the same speed as the external lines. However the HoL problem condemns the switch throughput [18]. The combination of input and output queuing – although improving the switch’s performance – needs complex scheduling algorithms. An alternative solution consists on introducing small buffers to the classically bufferless crossbar. The concept was first introduced in [33], and evolved with many propositions and

¹Prabhakar *et al.* [32] showed that a speedup of four is sufficient to emulate an output-buffered switch with output FIFO queues using a CIOB switch with VOQ queues.

modifications. The approach suggests storing cells in internal crossbar buffers before they are transferred to their outputs line cards. Buffered crossbars make the scheduling mechanism more relaxed as it uses simple and distributed arbiters. Some of the scheduling algorithms have been proposed [34–37] jointly with buffered switch fabrics. Yet, a fully buffered crossbar fabric introduces N^2 buffers (one per crosspoint) which makes it highly expensive and less appealing for large valency switches [36]. Therefore some works have been done on the partially buffered crossbars. This type of fabric has got only few internal buffers B (where $B \ll N$) [36], and it exhibits attractive features due to its similarity to the bufferless crossbar – in terms of cost – and to the buffered crossbar – in terms of performance.

Packet-switches design has gone through extensive research and innovation. It is possible to classify the IP switching architectures based on several criteria, as it will be shown in what follows.

2.3 Switching architectures

The first generation of IP routers were software-based, implemented on general-purpose Central Processing Units (CPUs). The processor and interface cards used to be connected among each other via a shared bus, and a centralized data memory was used to buffer data [38]. In a software-based design, data needs to cross twice the shared media; one time for processing, and later on for transmission. The centralized processor also executes the routing protocol, maintains the routing table, and performs the routing table lookup and control protocols. Using a central CPU, makes the processor a bottleneck for the router. Actually, the floating traffic load limits the CPU capability. At some point, it contributes to considerable performance degradation. Additionally, data transportation and table lookups are processing cycles greedy; which leads to *very* long transmission time. All in all, the software-design approach was proved to be unsuitable for high-speed routers [38].

2.3.1 Single-stage switches

There have been many ways to design IP routers. The switching fabric that makes the core of the router has attracted particular attention through time, and many trials have existed to improve upon previous solutions, and to cope with their limitations.



A little background: Time division fabrics

Shared memory fabric

A shared memory – typically – consists of a single dual-ported memory to which are connected all input and output lines [39]. Incoming packets are multiplexed into a single data stream. They are sequentially written into the shared memory as shown in Figure 2.3. Packets are organized into separate output queues inside a memory, one per output port. A memory controller is responsible for packet headers processing. It decides the order in which packets are read out of the Random Access Memory (RAM) at every time step. At the outputs, the flows of packets are demultiplexed.

The shared memory routers are known for their excellent QoS performance [40]. It is possible to achieve high throughput, since there is no contention as long as enough memory is provided. However, there are some restrictions for this architecture. First, packets must be written and read out from the memory at one time in every $\frac{1}{N \cdot S}$ seconds, where S is the port speed. As the access time of RAMs is physically limited, the shared memory architectures cannot scale up to large sizes and fast speeds. Moreover, the memory size is limited by the total bandwidth of the memory controller's Application-Specific Integrated Circuit (ASIC) [41].

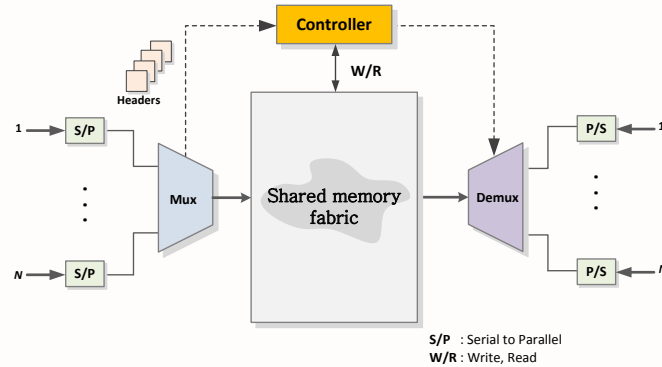


Figure 2.3: An example of a shared memory fabric [42].

Besides, for data consistency, shared memory switches require memory controllers running at four times the clock rate [43]. Then, to support a bandwidth of N connections, the memory controller's normalized I/O pins capacity should be $4N$ (for N input connections, N output connections, N memory write operations and N memory read operations). This limits the capacity of the ASIC that can be built, and makes the overall system implementation heavily dependent on the ability of the semi conductor industry to carry on creating faster memories and maintaining reasonable pin count while dissipating the minimal power at high frequencies [40]. There are some solutions to partially resolve the aforementioned problems, such as proper memory sharing strategies and high-speed scheduling – which is particularly required to maintain differentiated QoS [44]. In addition to the restricted scalability mentioned above, shared memory fabrics are fault-intolerant. Obviously, non backed up faulty fabric results in losing the communication between any input/output pairs. Furthermore, the design is HW inflexible since expansion purposes can only be satisfied by using a new fabric of higher capacity [42]. Some of the commercial routers that follow the shared memory approach in their switching fabric design, are Juniper Networks E-series/ERX edge routers and M-series/M20, M40 and M160 backbone routers[45, 46] as well as Cisco Catalyst 8500/8510/8540 multi-service ATM switch routers[47].

Shared medium fabric

In a shared bus fabric design, inputs and outputs are simultaneously connected to the same bus. The design is used with the Time Division Multiplex (TDM) technology. Figure 2.4 is a simplified view for a shared bus fabric. Each input

is allocated a time-slot to transmit data. Packets that come from all input ports are multiplexed and sequentially broadcast into a single high-speed bus with bandwidth $\geq N$ times the external line rate [39]. Each output port has an associated buffer and an address filter that examines the broadcast packet to see if it is destined for the corresponding port; in which case it redirects it to the associated buffer [42].

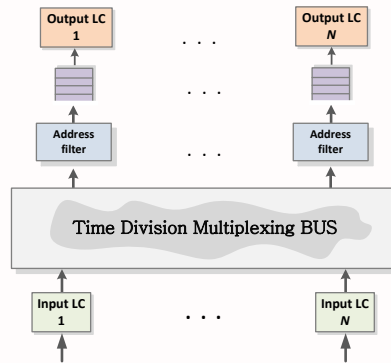


Figure 2.4: An example of shared bus fabric [48].

Both the address filter and the buffer need operate at the same speed as the shared bus. The shared bus switching fabric demonstrates different advantages that mainly report to the use of the simple and flexible access protocol. Moreover, the broadcasting aspect of packets transmission, allows uniform access among the input ports. It also facilitates the embedding of tree-based algorithms that facilitate the implementation of multicast. All the same, the port speed is a serious bottleneck in shared medium switching fabrics in general, and the shared bus in particular. As for the shared memory, the HW strongly constraints the scalability of a shared bus design. Both a large number of input/output pairs and a skewed traffic cause the performance *collapse*.

Output buffered fabrics with N^2 disjoint paths

The next step in the switching fabrics design, brought in use N broadcast input buses to which are connected N multi-access output buses, forming a fully connected topology [48]. As depicted in Figure 2.5, this design alternative provides independent paths between the N input/output pairs using a total of N^2 distinct and independent HW medium. An output consists of N address

2.3 Switching architectures

filters and N buffers; one for N input lines. As in a shared memory fabric, the address filters check out the broadcast packet headers on each set of N input buses to find packets which output destinations match those of the corresponding output lines. Next, packets are transferred to the associated buffers where they await before exiting to the output line.

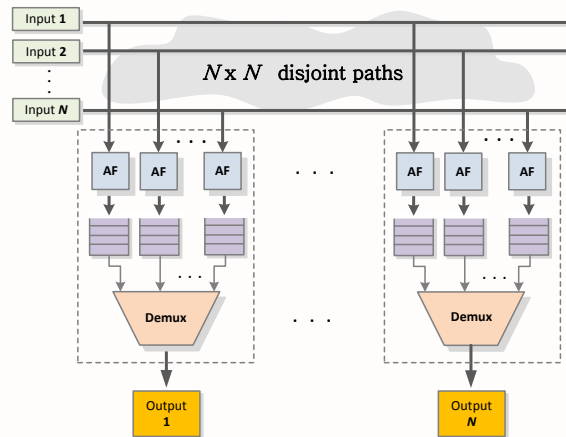


Figure 2.5: An example of output buffered switch fabrics with N^2 disjoint paths [48].

There are several advantages for the fully connected fabric. Given the independent nature of the paths, no input or output contention occurs. Moreover, address filters and output buffers are simple to implement. They do not require mechanical speedup, and only need to operate at the port speed. Since all hardware operates at the same speed, the fabric design can easily scale in size and speed. Moreover, the broadcast buses simplify the implementation of multicasting. The design is functionally output buffered, which means that it is theoretically capable of achieving optimal throughput. Nevertheless, the architecture involves intensive HW resources and the number of address filters and output buffers grows quadratically with the number of input/output ports, limiting the practicality and the expansion of the fabric design.

2.3.2 Space-division switch fabric: Crossbar

Early days of telecommunications have announced a new pattern to build interconnects. The crossbar switching fabric is made up of a matrix of size $N \times M$ crosspoint switching elements where N is the number of input ports, and M is the number of output ports, as shown in Figure 2.6. The crossbar is an old and tested design that provides a set of discrete and unique paths for each line card to transmit and receive data. Each input has one and only one dedicated path that leads to a given output port through a single switching point. It allows full bandwidth for any input/output pair of ports, in contrast with the bus and the shared memory based switches. A crosspoint switching element consists of a transistor that is geometrically located at the intersection of an input line and an output line. Each switching element gets one of the states ON/OFF depending on the output of the crossbar configuration phase. Interestingly, the crossbar is internally non-blocking. Hence, it is possible to simultaneously establish up to C connections at any moment of the time; provided that the C input/output pairs are disjoint (*i.e.*, no input/output pairs are blocking any other pair as long as the inputs and outputs between them are different), and that $C = \min\{N, M\}$. The crossbar fabric is also a fault-intolerant switching architecture. Any faulty part in the crosspoint switching elements in a given path, breaks down the connection between the associated input/output pair. Furthermore, crossbars suffer a fundamental limitation: Lack of scalability. Actually, the number of switching elements grows quadratically with the port count, making the switch capacity bounded to few Tbps – in the best case. This restriction is mainly attributed to the current available VLSI technology that still inhibits large single-stage crossbar switches implementation. The performance of the crossbar switch heavily relies on the packet buffer strategy and the packet scheduling process. Therefore, other design alternatives have been considered.

2.4 Multistage switches

Crossbar switches have the nice feature of being non-blocking. This means that the switching fabric, by itself, do not block the packets transfer, and that any

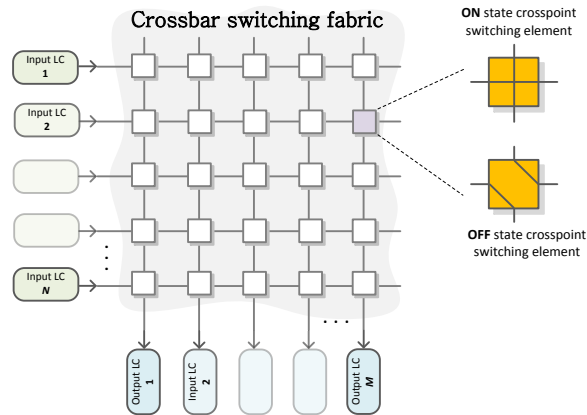


Figure 2.6: *Crossbar switching fabric.*

two idle terminations of the switch, should be able to establish a connection to ensure the end-to-end packet transmission. Most commodity switches/routers are conceived with crossbars (Cisco Nexus 5000 Series [49], Cisco 12000 Series Internet Router [50], Cisco Catalyst 5500 and 6500 Series [51], Juniper QFX10000 Switches [52]). The major inconvenience of single-stage crossbar switches, is their very evident limitations when used for heavy applications and large number of input/output ports. Beyond 32 or 64 ports, single-stage switches become unpractical and prohibitively expensive [53]. Communication systems engineers have solved the problem by using multistage interconnected switching elements to deliver partially or fully non-blocking architectures. The multistage schemes are implemented in different forms depending on a design requirements, predictability of connection, and the overall cost of the system. Multistage Interconnects (MINs) are built with switching modules of smaller radix and lower cost. They offer good broadcast and multicast features. They can also be build in redundancy and be incrementally expanded by adding more modules to an existing interconnect without disrupting the architecture.



Some interconnection network concepts

Multistage interconnects are diversified. Often, identical crossbars are used to build the network. However, some approaches adopt a mixture of switching fabrics, since it might be cheaper to manufacture without spoiling the targeted performance. Few classes of MINs that have been largely described and discussed in the literature are shortly overviewed.

Banyan networks

The Banyan network of Goke and Lipovski [54] is made of many crossbars interlinked in a way that pretty much looks like an eastern Indian fig tree; and so it was named after it. It comprises J stages, also called levels, where outputs of one stage directly connect to inputs of the next stage. The Banyan network levels are regular, if the number of inlets and outlets of crossbar modules are identical, or irregular, otherwise. The Banyan network provides one and only one path for every input/output tuple. Unfortunately, the architecture is internally blocking, and its performance collapses as the network size increases [55]. To cope with the internal blocking, buffered Banyan networks were introduced [56, 57]. Also, assembling many Banyan networks in parallel dilates the system, and provides multiple internal paths between the switching modules [58].

Omega network Ω

Shortly after the definition of Banyan networks, the Omega network was introduced as an isomorphic form of the Banyan network. An $N \times N$ Omega network, consists of $\log_2 N$ identical stages interconnected using a *perfect shuffle*. The inter stage connection pattern is uniform, and there is a unique path between any input and output modules. The Omega network was the first multi-stage architecture to demonstrate the self-routing property. Packets routing uses the destination addresses as tags, and happens in a distributed manner. Although the simple routing process, Omega network exhibits internal blocking, which reduces its performance.

Delta network Δ

The Delta network is referred to as *permutation* network, and sometimes called a *shuffle*. It was proposed by Lawrie [59]. The Delta network is a subset of the class of regular Banyan networks in which identical – but not necessarily squared –

crossbars are interconnected. Every stage of the network comprises binary switching elements, where one and only one state is available at a time (cross or bar). The inter-stage connection allows for a unique path between any input/output pair, which makes the network self-routing [60]. One bit indicating the destination is used to route a packet to the higher or lower output of its switching module. Same as Omega network, the Delta network is internally blocking.

2.4.1 The Clos-network

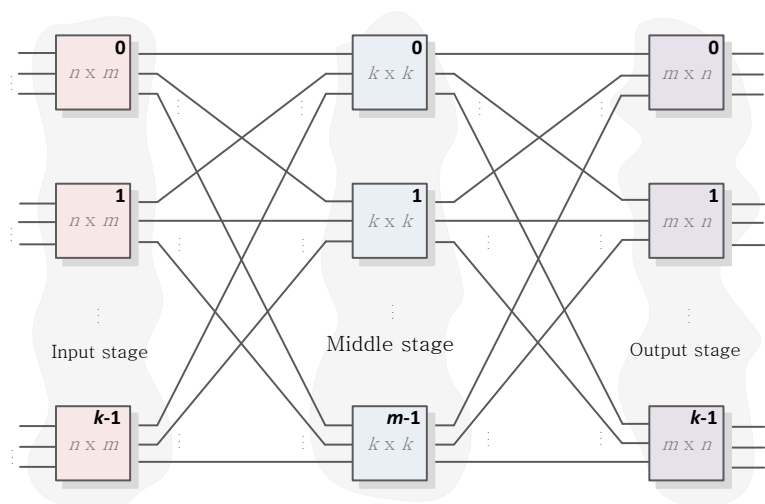


Figure 2.7: A three-stage Clos interconnect.

Figure 2.7 depicts a view of a three-stage Clos interconnect. The Clos-network was developed to satisfy the needs of the telephone switching industry, and to build a non-blocking network that uses fewer crosspoints than the crossbar network of an equivalent size. Backtracking to 1953, Charles Clos, published a seminal work in which he established the mathematical requirements to achieve a non-blocking multistage interconnect using interlinked crossbar modules. For $N = 36$, Clos suggested the use of \sqrt{N} crossbars at the ingress and egress stages, each of which is of size $\sqrt{N} \times \sqrt{N}$. The configuration involved 1188 crosspoints, to be compared to 1296 total crosspoints inferred in an equivalent crossbar [61].

Overall, the three stage Clos-network proved to have fewer crosspoints than the analogous crossbar network for all $N \geq 36$, and the growth factor of crosspoints in large Clos-networks is estimated to $\mathcal{O}(N^{1.5})$.

For a small network size ($N < 36$), Clos proved that it is better to opt for the crossbar interconnect, as it has fewer crosspoints. This fact reinforces the notion that multistage networks are best suited for large valency systems. Further in the same work, Clos has shown that the network is strictly non-blocking if the condition $m \geq 2n - 1$ holds¹. The five stage, seven stage Clos-network, and so on, can be build by adding more stages to a central three-stage interconnect. Based on the pattern, the dimension of the crossbar modules will change, but a rule of thumb is to extend the number of stages whenever the system size increases, to reduce the number of crosspoints. A key observation, is that Clos-networks exploit the power and flexibility of distribution, by dividing the functionality of a single crossbar over many smaller crossbars.

Lots of work has been done on multistage Clos-network packet-switches. Broadly speaking, these architectures can be categorized based on their buffer allocation schemes. The simplest Clos-network fabric to come across in the literature, is the bufferless one. Still other sophisticated alternatives have been investigated.

Space-Space-Space switch

The Space-Space-Space switch is quoted as S^3 switch. It has no packet buffers at any of its stages. It exhibits simple hardware. The real cost of an S^3 switch is inferred by the packet scheduling process [62]. However, the pure space network has the worst performance [63].

¹The notation n is used to denote the number of input ports and output ports per IM and OM, respectively. Also, the term m is employed to refer to the number of CMs in the three-stage Clos-network architecture.

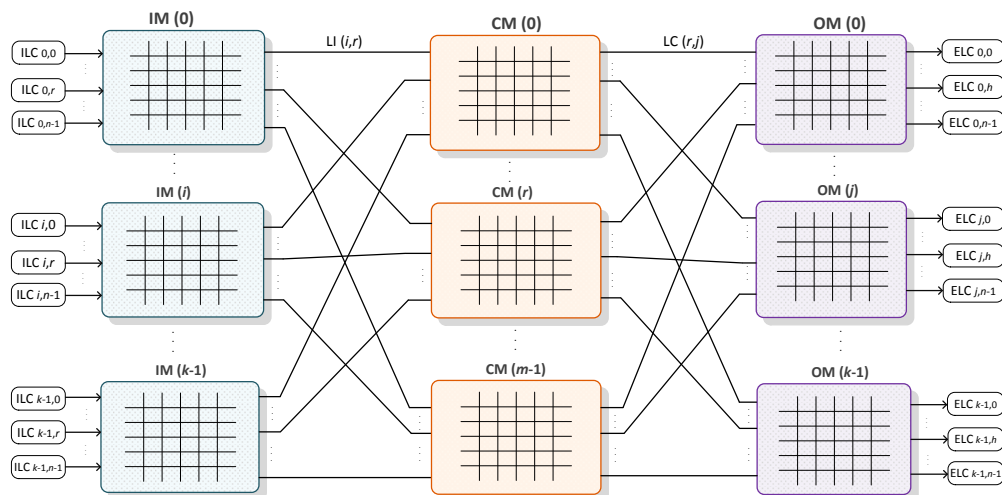


Figure 2.8: *Space-Space-Space Clos-network switching fabric.*

Contention happens at all points of the bufferless network. Packets compete drastically to exit the switch, and go to their output ports. Conflicts are to be constantly resolved using an adequate scheduling mechanism. Consider a Random Dispatching (RD) process of packets (*i.e.*, packets randomly choose to go through a particular central module), the S^3 switch with an expansion factor of 1.0 has a maximum throughput $\approx 39.7\%$ under uniform traffic. To achieve 100% throughput, the bufferless network needs a minimum expansion factor of 1.5 when the switch size is large [64]

👉 **The expansion factor** is an important parameter for Clos-networks. It is given by the ratio m/n , with n being the number of input ports (output ports) per Input Module (IM) (Output Module (OM), respectively) and m being the number of CMs. The expansion factor reflects the number of middle stage switching modules available to use. In the general case, the number of central modules is greater or equal to the number of ingress/egress ports of the first and the third stages of the network (this has to do with the necessary and sufficient condition for a Clos interconnect to be strictly or rearrangeably non-blocking. See Table 2.1). Raising the expansion ratio, increases the internal bandwidth, and – hence – reduces the blocking likelihood of the interconnect. Yet, it is difficult and cost-prohibitive to implement high-speed multistage switches with large expansion ratios, and so, alternative solutions have been considered to improve the performance of Clos-network based switches.

Table 2.1: Comparison between the different switching architectures [65, 66].

Parameter Type of the network	Type of cells	Number of Cells	Latency of any path	Blockingness*	Comments
Crossbar	Crosspoint	N^2	$\log_2(N)$	non-blocking	Widely used in the context of communication and switching
Banyan*	$d \times d$ switching elements	$\frac{N}{d} \log_d(N)$	N	Internally blocking	Generally used in combination with other network or to mount parallel networks in order to resolve the internal blocking issue
Delta*	$d \times d$ switching elements	$\frac{N}{d} \log_d(N)$	$\log_d(N)$	Internally blocking	Delta networks have been modified to introduce multiple paths through the network and to increase the reliability and the throughput
Omega*	$d \times d$ switching elements	$\frac{N}{d} \log_d(N)$	$\log_d(N)$	Internally blocking	Subset of Delta networks which form is a special case of the shuffle-exchange networks. An advanced mathematical tool in the sorting and interconnection networks
Three-stage Clos*	Varies	$2Nk + \frac{kN^2}{n^2}$	$\geq (2n-1)(2nk+k^2)$ $\geq 2kn^2 + nk^2$	Strictly non-blocking Rearrangeably non-blocking	Symmetric $\iff m \geq 2n-1$ Symmetric $\iff m \geq n$
Benes	2×2 switching elements	$N \log_2(N) - \frac{N}{2}$	$2 \log_2(N)$	Rearrangeably non-blocking	A special case of the Clos-network. Widely analysed and used for its simplicity and practicality

*. For a squared network
 ♣. The symmetric three-stage network is made of k crossbars at the first and third stage, each of which is of size $n \times m$ and $m \times n$, respectively. The middle stage contains m switching modules, each of dimension $k \times k$.

It is important to understand contention, since a great part of the previous and on-going work in the switching design area (in addition to hardware design), focusses on resolving the perpetual contention [67, 68]. Here is a summary of the contention points where packets conflict for the next shared link or fabric.

 **Where does contention happen in S^3 ?**

- Packets first compete at the input port to access the switching fabric (see Figure 2.9). Up to N packets coming from N VOQs associated to the Input Port $IP(i, r)$ may arrive at time-slot t . However, only one packet is allowed to enter the fabric at a time*.

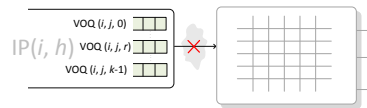


Figure 2.9: Contention at IPs.

- Once in the first stage fabric, a packet contends with others – packets – to access a central stage module through a shared link bridging an IM and a CM as illustrated in Figure 2.10.

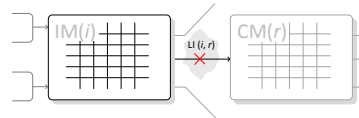


Figure 2.10: Contention for CMs.

- Figure 2.11 depicts the last contention stage that takes place at the middle stage of the Clos-network, where a packet tries to win access to the desired OM where resides its corresponding Output Port $OP(j, h)$.

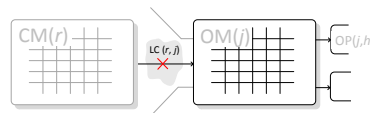


Figure 2.11: Contention for the OPs.

*Assuming no speedup is used to run the inter-stage links, and that links run at the same line rate as external lines.

In a similar way, contention happens in other types of Clos-network architectures, whenever two or more packets request a common resource as they make their way throughout the multistage fabric.

Memory-Space-Memory switch

The MSM architecture have attracted attention in the research area [10, 11, 13, 64, 69–71]. It has been introduced to scale the switching architectures, and to reduce the blocking rates by involving reasonable cost and complexity [69, 72]. The design approach is quite similar to that of the single-stage CIOQ crossbars switch. More importantly, the work of emulating an OQ switch with an CIOQ switch can be extended to MSM [17, 73]. Input and output buffers have been added to absorb contention at – both – the first and the last stages. Adopting bufferless central modules is promising, since even at high-speed port rate, packets still can reach their output ports in order [64]. Now that no buffers are meant to accommodate packets in the middle stage, the real challenge is to efficiently sort out the contention, and to transfer packets between the network stages. Usually, this is done through a packet dispatching algorithm. The algorithm does not only elect packets to be served within an input queue (choosing one from the VOQs of the input port to serve its HoL cell), but also appoints the end-to-end path from an input switching module until the appropriate output module. The MSM switching architecture – despite its interesting features – relies on centralized scheduling algorithms to assign paths to pre-selected VOQs. The bufferless central stage of the Clos-network implies the necessity for a centralized packet scheduling mechanism since no intermediate buffers are deployed to temporary hold transiting packets. Therefore, a central arbiter is mandatory to control the set of available resources at a time, and to find out a conflict-free matching between requesting VOQs and the output ports.

Memory-Memory-Memory switch

Previous works have shown that introducing memories to all stages of the Clos-network is liable to improve the switch performance [74–77], and to offer high throughput. In fully buffered multistage switches, the packet scheduling process is generally simple and dispersed, as contention is entirely absorbed by distributed buffers. However, a major concern of the MMM switches, is the

2.4 Multistage switches

out-of-sequence packet delivery. Adding buffers to the middle stage of the Clos-network makes packets of the same flow experience variable queuing delays. This – consequently – leads to mis-sequenced packet transmission [12, 78]. Hardware wise, an MMM switch is cost prohibitive, which has urged the need for less complex and less expensive solutions that tangentially perform as good as an MMM switch.

The Space-Memory-Memory (SMM) switching architecture as described in [79], is simpler to implement than the MMM switch. It requires no scheduler at any network level or module. Instead, a static connection pattern has been used in the input stages, and self-routing has been implemented to transfer packets to the subsequent two stages of the switch. In later works, the authors in [62] have proved that the SMM switch with a Desynchronized Static Round-Robin (DSRR) connection pattern in IMs, is liable to achieve 100% throughput under admissible traffic. In Table 2.2, some examples of packet-scheduling algorithms

Table 2.2: *Examples of scheduling algorithms for different switches.*

Switch type	Arbitration policy		
	Random	Round Robin	Static dispatching
Crossbar	PIM [80]	iSlip [21]	SRR [81]
S^3	-	-	Distro [71]
MSM	CD [69]	CRRD [64] CMSD [64] PCRRD [82] CRRD-OG [70] MWMD [83] MOMD [83] FRD [84]	SRRD [85]
SMM	-	-	DSRR [62] modified DSRR [§] [79]
MMM	RD	RR [§] [86]	-

§. Distributed scheduling in which a combination of DSRR at the IMs and OCF at the OMs is adopted.

†. For packet dispatching in the Trueway switch.

that have been suggested for the single-stage crossbar switch, as well as the MSM, SMM, and MMM multistage switches are given.

2.4.2 In-sequence packet delivery

In multipath network switches, the existence of buffers in the middle stage of the switching fabric, causes packets to reach the last stage – and consequently their output ports – out-of-order. Packets are likely to experience variable queuing delays when they are routed through different paths with varied queue lengths in the fabric [12]. The state-of-the-art solutions for the out-of-order packet delivery is diverse. Resequencing buffers have been typically used at the output ports to re-establish the order of packets that originally belong to the same flow. The approach is about maintaining earlier packets buffered at small resequencing buffers, until it is their turn to be served with reference to their flow’s number of sequence. A time stamp based approach [87–89], marks every packet with a Sequence Number (SN) at its arrival to the input port. Next, the SN serves next to correct the order of packets. The time stamp resequencing is simple to implement. However, it reaches its limitation with large-scale switches, where the time overhead becomes larger than practical [89]. With larger overhead delays, comes a higher cost and complexity that only can be reduced at the expense of a lower performance [90]. Counter/sequence number resequencing suggested in [91, 92] are highly efficient. Yet, they incur high hardware complexity, and the resequencing buffers can grow indefinitely, if the extent of mis-sequenced packets is not controlled. Window-based resequencing is similar to the time stamp approach [90]. Every flow is assigned a SN that range from 0 to $W - 1$, with W being the size of the window. Such a measure ensures that the number of packets per flow inside the switch fabric does not exceed W [93]. Consequently, no feedback control signalling is required. So far, window-based resequencing affects the throughput performance. Therefore, authors in [78] proposed an iterative matching process at the last stage of a three-stage Clos-network switch, that improves the switch throughput and forwards packets in order of their arrival. However, the algorithm is complex and time-consuming. An extensive number of works stated that it is possible to preserve packets order by steadily forwarding packets of the same flow through the same path [86, 94]. Hash solutions rely on different hash functions to map flows to paths in a way to distribute the traffic, and to minimize congestion. In

practice, the static hashing scheme do not deal with the variable and heavy flows. Even when the flows are evenly distributed among the available routing paths, the wavering flow sizes can cause local congestion and – sometimes – bandwidth underutilization. Dynamic hashing methods [95, 96] arose to smoothly adapt flows distribution according to the network’s traffic load. Status tables are maintained to check the availability of a given link as well as points of failure in the network. These tables are updated with reference to the congestion status of the network. Unfortunately, dynamic schemes are too complex and infeasible for large-scale switches. Far away from corrective actions, a centralized packet routing is an effective way to forward packets in order. Yet, although successful for small-scale switches, collecting information from all switch components in large-scale switches, is complex and impractical [12, 97]. The packet scheduling logic is key in the design of high performance packet-switches. It works tightly with the switching architecture to manage the packet transfer from the input line cards to the corresponding output line cards. In the following, the classes of packet scheduling algorithms are overviewed, and the difference between them is stated. Examples of well investigated scheduling schemes are also given.

2.5 Packet scheduling

Designing a switch goes through a structured process that considers three main components: The switching architecture, the hardware requirements, and the logic to schedule packets. The process of packet scheduling is actually part of the architecture design. It is the way to manage packets transmission across the switch. It also defines the service policy among packets, and allocates scarce link capacity among contending flows [98]. The allocation regulates an application – or in the general case, a network – objectives. There is a replete research literature on novel packet scheduling algorithms that minimize the Flow Completion Time (FCT) [99–101], flexibly allocate the available bandwidth, work on promptness or space buffer across flows, and use the Weighted Fair Queuing (WFQ) [102], etc. A focus is put on the scheduling algorithms that were conceived for high-performance, high-speed switches. Undoubtedly, a proper scheduling algorithm proposed jointly with the adequate switching architecture,

is key to enhance the performance by maximizing the resource utilization through statistical multiplexing of packets. In a regular $N \times N$ crossbar switch with VOQs, the scheduling problem can be formulated as a bipartite graph matching. Denote $Q_{i,j}$, the state occupancy of $VOQ_{i,j}$ at time-slot t ($1 \leq i, j \leq N$). At the starting of every time-slot, a graph $G(V, E)$ ¹ with $2N$ vertices is constructed where N input ports and N output ports are involved. The set of edges is made of the connections between input i and output j for which the corresponding $Q_{i,j} > 0$. A valid schedule connects an input i to at most one output j at a time-slot, satisfying the bandwidth restrictions of the crossbar [22]. Afterwards, the matching problem can be solved using the *bipartite edge colouring*. The basic idea is about assigning a colour to each edge in the set E , such that adjacent edges are allocated different colours. The resulting set of minimum number of colours used to mark the edges for any graph is called *chromatic index of the graph* [103], and the time complexity of finding a perfect matching in an $N \times N$ bipartite graph is $\mathcal{O}(N^{2.5})$ [104].

It is essential to distinguish between a *maximum* and *maximal* matching. A maximum matching is absolutely the largest matching size that can be made on the graph. It represents the optimal matching. In the real world, suboptimal matching also known as maximal matching can come to replace a maximum matching. The maximal matching is the one to which no further edges can be added without altering the already matched set of vertices. Thus, any maximum matching is maximal, but the opposite is not valid. The scheduling algorithms can be broadly categorized into weighted and non-weighted [19]. Next, two different classes are discussed: Maximum size matching and maximum weight matching types of scheduling algorithms. A matching has maximum size if the number of edges is maximized, while a matching has maximum weight if its weight is maximized.

¹ V is the set of vertices, and E is the set of edges of the graph.



Matrix representation of the matching problem

Scheduling the set of input and output ports of the crossbar switch can naturally map to a binary *Request Matrix* $\mathcal{R}_{N,N} = [r_{i,j}], 1 \leq i, j \leq N$, where $r_{i,j} = 1$, if there is an edge between input i and output j in the graph (*i.e.*, whenever there is a packet in input i that requests output port j), and $r_{i,j} = 0$ otherwise [53]. Resolving contention is equivalent to finding the one-to-one matching, for which at most one request can be selected per input (row) and per output (column). It suffices to find the subset of requests $\mathcal{S} \subset \{r_{i,j}\}$ that simultaneously satisfy the following constraints:

$$\sum_{(i,j) \in \mathcal{S}} r_{i,j} \leq 1, \quad \forall i : 1 \leq i \leq N$$

$$\sum_{(i,j) \in \mathcal{S}} r_{i,j} \leq 1, \quad \forall j : 1 \leq j \leq N$$

The following example depicts the *Request Matrix* for a 4×4 crossbar switch as well as the maximum and maximal matching matrices.

$$\mathcal{R}_{N,N} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{S}_{\text{maximum}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{S}_{\text{maximal}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.5.1 Maximum size matching

As it was described before, a maximum size matching is the one that maximizes the number of edges in the subset of requests \mathcal{S} at every time-slot, so that to maximize the *instantaneous* throughput of the switch. In [105] and [104] were proposed two well-known maximum size matching algorithms where time complexity is estimated as $\mathcal{O}(N^{2.5})$. McKeown proved that a maximum size matching is liable to achieve full throughput under uniform *Bernoulli i.i.d* arrivals [19]. All the same, this class of matching can be unstable and unfair under admissible traffic. In a much worse scenario and under inadmissible traffic, a maximum size matching algorithm is likely to lead to starvation [23]. The

high computational complexity of the maximum size matching algorithm renders it unpractical. Often, heuristic algorithms are proposed. These algorithms are practical, and they may be considered to approach the performance of the optimum matching with variable effectiveness and complexity (*e.g.*, the maximal size matching class algorithms).

2.5.2 Maximum weight matching

The weight of a matching is defined as the sum of the metrics that have been assigned to the edges included in the subset \mathcal{S} . The weight is chosen to be some quantity that reflects the level of congestion. A matching is heavier another, if its weight is greater than the other one. Originally, weights were the length of the VOQs. However, mean queues lengths are known to change slightly through time which implies that a heavy matching is likely to remain heavy for a few or more time slots [106]. In some works [23], the age of the oldest packet in the VOQ was used to weight edges of a matching. In general, a maximum weight matching is stable with 100% throughput under all admissible traffic [23, 107]. Moreover, MWM performs optimally in terms of delay [108]. Some algorithms have been proposed to provide exact delay bounds [17, 32, 109]. Yet, MWM algorithms are impractical because of their high time complexity (in the order of $\mathcal{O}(N^3 \log(N))$). This makes them too complex and too slow for large high-speed switches.

2.5.3 Practical maximal size matching

The common resort for packet scheduling in large scale and high-bandwidth switches, is to adopt maximal matching. There is plenty of literature that discussed and evaluated performance of maximal size matching algorithms [23, 23, 28, 80, 110, 111]. The maximal matching algorithm finds a matching subset to which no further edges can be added without removing an already matched edge. On the average, a maximal size matching algorithm requires $\log_2(N)$ iterations to find a maximal matching. The first observation, is that the complexity of this type of scheduling significantly rises with the port count, which basically explains the struggle to implement a maximal size matching in large

valency and high-speed switching. Some works impose less constraints on the QoS or use static priorities and RR scheduling to come up with implementable solutions [112]. Maximal size algorithms perform the matching in three steps known as Request-Grant-Accept handshaking protocol (see Figure 2.12).

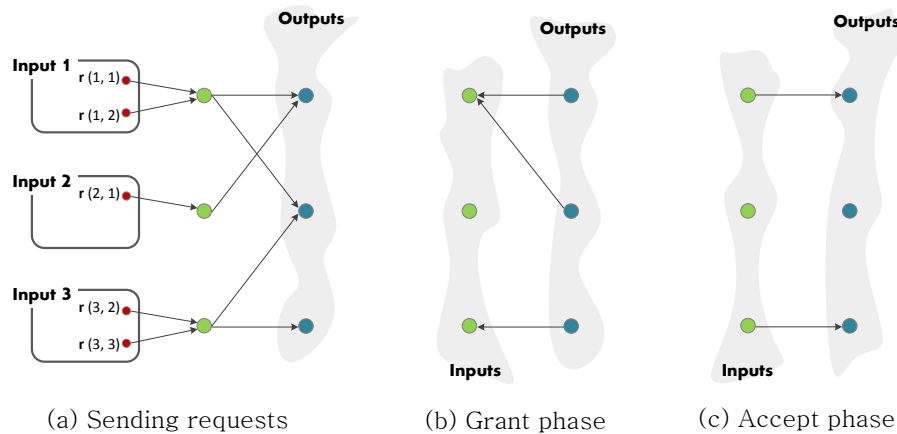


Figure 2.12: An example of a Request-Grant-Accept matching used in the PIM algorithm [80]. Unmatched inputs send requests to outputs for which they have a packet. Outputs grant one of the received requests, and ties are broken randomly. Input ports also select one of the returned grants in a random fashion. Note that at the end of the first iteration, input 2 remains unmatched, although it does not conflict with any other connection. It would be matched at the second iteration [19].

The PIM is an example of a practical maximal size matching algorithm that was first proposed in [80] and adopted by DEC Systems Research Center¹. PIM formed the basis for a plethora of algorithms, which all run the same three steps matching, but have different scheduling criteria. The Request-Grant-Accept concept was also adopted by McKeown in the iSlip algorithm [21] used in Cisco routers.

¹In 1993, DEC Systems Research Center, Palo Alto, CA, developed the experimental 16-port ATM LAN switch that they called AN2. The switch has a capacity of 1Gbps and was commercialized as the Gigaswitch/ATM

Request-Grant-Accept protocol

- Each unmatched and non-empty input queue sends a request in the name of its HoL packet to the corresponding output port. The default weight given to any request in a maximum/maximal size matching is 1, whenever the input queue is non-empty. Weighted algorithms add weights to requests as a way to prioritize the longest queue first (iLQF[19]) or oldest cell (iOCF [19]), and so to maximize the weight of the matching.
- Each output reckons up the received requests and grants one of them according to a particular logic that varies from an algorithm to the other. For instance, PIM [20] randomly selects one of the requests. The iSlip algorithm sends a grant to the request which priority currently appears in the RR selector. The grant pointer of iSlip [21] is not updated unless the grant is accepted. This is the essence of the iSlip algorithm that made pointers desynchronize and – consequently – the switch throughput get higher. The Static RR (SRR) [81] algorithm starts first by initializing its grant pointers in a fully desynchronized manner. Thus continuously updating pointers maintains the desynchronization effect.
- Upon receiving the grants from output ports, inputs accept one of them using the same logic adopted for the grant phase, or using another selection pattern (by considering the weight of the grant).

Multistage networks have been long ago used for the packet-switching design. The topic is generally well investigated. However, there is always a room to make the performance better, and the packet scheduling more efficient. Recently, Networks-on-Chip have been suggested to re-design the fabric of switches/routers. The objective is to probe new ways that – probably – can outperform the classical proposals, and meet the set of design requirements including feasibility and high-performance. In this chapter, common NoC types are loosely presented. The packet switching and routing, as well as congestion-control mechanisms are described. In subsequent chapters, more details about these aspects to formulate choices for the switching architectures and the scheduling algorithms are provided.

2.6 Networks-on-Chip: Concept for IP packet switching

Networks-on-Chip have emerged to meet the growing computation intensive and power hungry applications. With advances in the VLSI technology, it has been possible to support an intense integration of transistors announcing the early days of NoCs. In most System-on-Chip (SoC) applications, interconnection between the computing resources was an issue [113]. Conventional bus connections with arbitration schemes to manage access requests, were adopted because they are rather simple than cheap. The bus switching approach is flexible, but it allows only one communication transaction at a time. Nodes share the system bandwidth making the scalability of the connection pattern a serious bottleneck. Using many bridged buses or hierarchical bus cores can partially solves the bandwidth requirement, and ameliorates the communication parallelism [114]. Yet, the shared bus connection is unsuitable for large scale, large bandwidth systems [115]. The NoC architecture has a potential throughput that is approximately 10 times that of a bus-based architecture [114]. This ratio can be lowered using a multi-layered buses architecture, which is quite similar to a crossbar. But the added complexity limits the target frequency anyway. Table 2.3 and Table 2.4 overviews some differences between a shared bus and NoC architectures, and give a rough idea about the asymptotic cost function of each of them, respectively. NoCs were proposed to support the thriving of SoC applications in different ways. Inspired by traditional large-scale multiprocessors and distributed computer networks, NoCs largely reuse the conventional communication concepts to boost the overall system performance. NoCs have different premises than off-chip networks. The major difference resides in the hardware constraints and synchronization. Actually, NoCs have stringent constraints since on-chip buffering and computation resources are relatively expensive [116].

2. SWITCHING AND CONGESTION-CONTROL: STATE-OF-THE-ART

Table 2.3: *Shared bus versus Network-on-chip* [114].

Criteria	Shared bus	Network-on-Chip
Type of connection	Multipoint synchronous	Point-to-point, layered, Globally Asynchronous Locally Synchronous (GALS), etc
Max frequency	250 MHz	>750 MHz
Peak throughput	9 GB/s (more if wider bus)	100 GB/s
Cluster min latency	6 Cycles @250 MHz	6 Cycles @250 MHz
SoC min latency	14-18 Cycles @250 MHz	12 Cycles @250MHz
System throughput	5 GB/s (more if wider bus)	100 GB/s
Average arbitration latency	42 Cycles @250 MHz	2 Cycles @250 MHz
Gate count	~400K	~210K

Results above are given for a NoC architecture with 4-byte wide links, and a shared 4-byte data wide bus architecture, evaluated in a cluster of 9 buses. Frequencies of 250MHz for the bus-based architecture, and 750MHz for the NoC-based are assumed in the comparison.

Table 2.4: *Asymptotic cost function* [117].

Criteria	Shared bus	NoC
Total area	$\mathcal{O}(n^2\sqrt{n})$	$\mathcal{O}(n)$
Operating frequency	$\mathcal{O}(\frac{1}{n})$	$\mathcal{O}(1)$
Power dissipation	$\mathcal{O}(n\sqrt{n})$	$\mathcal{O}(n)$

Moreover, the number of on-chip interconnection links is far larger than what off-chip communication networks require. Embedding RAM memory blocks on-chip is also expensive [118]. Still, it is possible to distribute the storage space over the NoC components such that small size buffers are implemented. However, this is not without problems, as the overhead area becomes dominant. From another perspective, the computation to be processed on-chip, is relatively much costly than the one to be performed in computer networks. The reason is that off-chip networks usually incorporate a dedicated processor to relieve the computation part from the communication processing. Such a solution do not apply for NoCs because the size of the network interface would be too large than admissible. Inter nodes links in NoCs are shorter than in off-chip networks. This feature allows tighter synchronization and smaller node buffers, since the communication is done at smaller granularity. The good news, is that the current semiconductor technology offers fast and reliable wires that help compensate for the lack of memory and computational resources. Table 2.5 gives points of similarities and differences between NoCs and computer networks.

2.6 Networks-on-Chip: Concept for IP packet switching

Table 2.5: *Points of similarities and differences between NoCs and computer networks.*

Similarities	Differences
Both are a network of nodes. A node can be a router/switch, link, Processing Element (PE), etc.	While computer networks are made for general purpose, NoCs are designed to target specific application domain. for general purpose.
Both use packet-switching.	Computer networks support plug and play router. However a NoC is planned by design and the placement is unchangeable.
The routing information is retrieved from the packet/flit header where are stored the source/destination address, error detection/-correction bits, priority, etc.	Power management is crucial in NoCs. Generally, NoC vendors provide tools to configure and adjust the on-chip interconnect to guarantee the necessary bandwidth while reducing the die area and power consumption.
Both implement specific communication protocols for traffic routing, arbitration and flow control.	NoC cannot support heavy communication protocols unlike computer networks.

NoCs are remarkably modular. However, designing a Network-on-Chip concerns with several issues, ranging from the topology to the routing scheme, performance and cost/complexity compromises. The topology can either be regular or irregular. Regular topologies are preferred for their interesting geometry and layout features. They can be easily build, reused and optimized using regular iterative blocks. Frequently, the notation k -ary n -cube is used to describe the NoC topology, with n being the number of dimensions and k being the degree of the dimension. Figure 2.13 depicts examples of some regular NoC topologies. Any-ary 2-cube is – simply – the 2 dimensional NoC, most often denoted as 2D NoC. This topology is the most preferred for its regularity, simplicity and

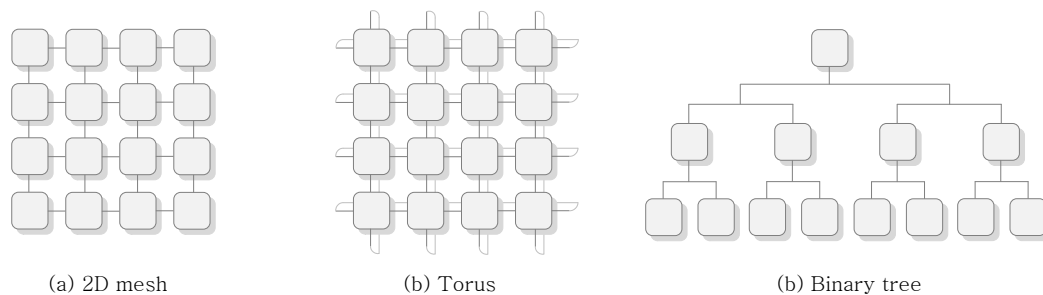


Figure 2.13: *Examples of regular Networks-on-Chip topologies.*

scalability. All the same, there are plenty of essays on different NoC topologies where has been discussed and evaluated [119], the performance of the topology and the on-grid routing algorithms [120]. The Torus topology is also a regular NoC. It is quite similar to the 2D topology except that nodes at an edge row (column) are connected to nodes at the other edge row through wrap-around channels. A k-ary binary tree with a central root from which flow connections to other leaf nodes in an order of hierarchy, is another type of regular NoCs.

NoCs provide communication concurrency and distributed resource sharing which necessitates arbitration [121]. Interestingly, on-chip routers are fitted with advanced features, such as pipelining, prioritization, allocation schemes, etc. They take the routing decision based on the local information, and transfer data packets from a source to a destination node. As for off-chip networks, the arbitration process impacts the efficiency of the network as well as the workload balance and the network latency, playing an important role in the whole system performance. A *good* algorithm should have fair routing path lengths, bearable worst-case packet delay, low average latency and good workload balance to assure high throughput [122]. Routing in NoCs has been a hotspot research area and lots of research conclusions have been made in various aspects. On-chip buffers can be small. They only require some flow-control techniques to make sure that any overflowing is prevented [123]. In the following part, the different switching modes often used in the context of NoCs are reviewed. A switching mode defines how data traverse its route throughout the network.

2.6.1 Packet-switching strategy

There are two main types of switching: Circuit switching and packet-switching. In circuit switching, data cross the network using a dedicated end-to-end path. The circuit can be physical or virtual. It is reserved before starting the transmission, and it remains in use until the data transfer is over. In packet-switching, messages are segmented into a sequence of packets that are separately conveyed through the network without prior path reservation, allowing better utilization of the network resources [121]. Three packet-switching techniques are recognized: The store-and-forward, cut-through and wormhole switching.



Packet-switching modes

Store-and-forward

Early direct networks used the *store-and-forward* switching technique [124]. As shown in Figure 2.14, a network node must wait for the whole packet to arrive before it forward it to a downstream node. The node verifies the integrity of the packet. Actions for error correction and table lookup are processed at every node. If everything is fine, then the packet is ready to be forwarded to the next hop. In the literature, the term *flit* is used to refer to the smallest data unit that can be transferred across a link, and that can be either accepted or rejected based on the return of the link-level flow control. The store-and-forward switching mode implies a non-contentional transmission latency that depends on the length of the packet, as shows the following expression.

$$T_{store\ and\ forward} = \left(\frac{L}{W} + T_h \right) \times N_h$$

With L being the total number of transmitted flits. W is the link bandwidth, T_h is the routing delay per hop and N_h is the number of hops from the source node to the destination node.

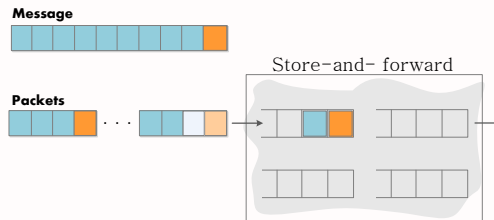


Figure 2.14: An example of the store-and-forward switching.

Virtual cut-through

The virtual cut-through switching was proposed to decrease the data transmission time delay [125]. A node of the network receives a portion of the packet and then forwards it to the buffer of the downstream node as soon as the corresponding resource is acquired, as Figure 2.15 depicts. Unlike the store-and-forward switching, nodes do not wait for the entire packet to become available

at its local buffer. Hence, the non-contentional latency (specified below) is lower than in the conventional store-and-forward [126].

$$T_{\text{virtual cut-through}} = \frac{L}{W} + (T_h \times N_h)$$

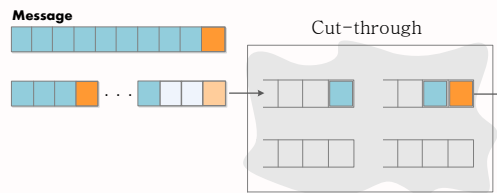


Figure 2.15: An example of the virtual cut-through switching.

Wormhole

The wormhole switching [123, 127, 128] works like the virtual cut-through switching. However, it combines flits transmission with data streaming. Flits travel the network in a pipelined fashion, crossing small buffers meant to hold a small proportion of the packets [129]. Flits that belong to the same packet follow one way throughout the network until their destination. The entire path is determined by the header flit which routing information is used to decide about the next hop. Other flits just follow – passively – the same itinerary as the head flit. If blocking happens, flits of the packet remain in place until downstream resources become free. Figure 2.16 gives an example on how flits progress in the network. The pipelined transmission makes the non-contentional latency transmission of L flits, the same as that of the virtual cut-through switching. It is almost independent of the number of hops, when there is no contention, and the packet length is large.

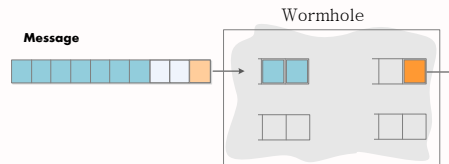


Figure 2.16: An example of the wormhole switching.

2.6 Networks-on-Chip: Concept for IP packet switching

NoCs have been introduced to scale down the concepts of large scale networks, and to apply them to the embedded SoC domain. *Yet, they are still networks¹!* They need a clear methodology to transfer data units from a source nodes to a destination node.

2.6.2 On-chip routing

Communication patterns in NoCs have been first borrowed from computer networks and adopted to on-chip communication needs (as it is summarized in Table 2.5). As in off-chip networks, contention and resource allocation are key routines that necessitate adequate arbitration schemes to be implemented. Packets acquire their path through the network thanks to a routing algorithm. In Networks-on-Chip, the routing decision happens at the node level. To save storage area², and lower the process complexity, routing in NoCs do not involve global state information of the network. The routing algorithms can be classified in terms of path diversity into *deterministic*, *oblivious* and *adaptive* routing [121].

- **Deterministic routing** is the simplest and most popular scheme for NoCs [130]. A path is completely determined using only the source/destination pair of address of the packet. A deterministic routing ignores the path diversity of the NoC, and blindly forwards packets of the same flow throughout the same path.
- **Oblivious routing** includes the set of deterministic routing algorithms. It considers a system of optional paths in advance for any source/destination pair. For instance, a random selection of paths uniformly distributes packets among different paths. This class of routing also ignores the geometry of the network and does not exploit the path diversity [124].

¹As it will be seen in subsequent chapters, NoCs are self-contained networks with all necessary blocks such as routers, buffers, embedded flow-control logic, etc.

²Using a global network information to route packets across the Network-on-Chip, would require that this information be shared with all nodes. This would absolutely call for additional storage in every on-chip router [129], and rise the implementation complexity of the system.

- **Adaptive routing** processes the calculation of the packet route. Routes are network-state dependent [131]. Both local (node level) and remote (network level) information can be used to take the decision about the packet's next hop. Paths are dynamically assigned taking into consideration some conditions such as congestion, fault presence, etc.

A routing algorithm can be minimal or non-minimal. A minimal routing selects the shortest available path bridging the source and destination nodes. Whereas, a non-minimal routing allows packets to follow longer paths if the shortest one is inaccessible. Non-minimal routing may result in larger latency than minimal routing. However, it better explores the network topology in case of congestion, and contributes to the blocking minimization.

2.6.3 Deadlocks

In networks, deadlocks are a serious issue. They are as viral as starvation since both can cause network freezing. Deadlocks can occur when a packet keeps on waiting for an event that cannot happen (*e.g.*, two or more packets are held waiting for the other packet to release a resource. In which case, all packets enter a circular wait condition.) [132]. Many proposals consider deadlock avoidance while designing the network topology. Still, cycles in buffers can also lead to deadlock situations, where buffers are full and no more packets progress through the network structure. Thus, it is important that the routing algorithm is made cycle-free. If it is not the case, then packets must be dropped from saturated buffers to make free space for pending cells.

The network can be sometimes loaded – a bit – behind its capacity (if traffic is accepted without clear restriction). Additional data could be eventually stored all the way to the source nodes. However, this would block the links in between, and degrade the network performance. In the following section, the importance of congestion-control is specified, and the difference between active and proactive control schemes is explained.

2.7 Congestion-control

Multistage packet-switching design brought about solutions to many limitations of the single-stage switches. Yet, these large systems – also – did rise other challenges. In addition to the network topology and routing algorithms, flow and congestion-control are two major concerns. Congestion-control is an open research problem. Without it, performance would severely be affected under real-world traffic patterns. In general terms, congestion-control is the set of mechanisms and techniques that tend to disperse the load in the network, and to keep it below the capacity [133]. Congestion in a network may occur if the number of influent packets exceeds the capacity of the network (equivalently, in terms of number of packets). It also takes place whenever a link, node or buffer is incapable of handling so much load, that the QoS *collapses* [112]. In literature, congestion is described as a state of network saturation that lead to the degradation of the vital network parameters (throughput and delay) as resources such as communication links, packet buffers and/or processor cycles reach their limit. The existing congestion-control approaches can be categorized into: Congestion avoidance and congestion recovery. This classification do not reflect the congestion type, nor does it provide details of the congestion management approach.

2.7.1 Reactive congestion-control

Reactive congestion-control [134–136] do not work to prevent congestion form happening at the first place. A node in the network recognizes congestion when its queue length reaches a given threshold. The Forward Explicit Congestion Notification (FECN) [137] is an example of a reactive scheme. It works as follows: On the onset of congestion, the congested node propagates the information by setting a congestion notification bit in the header of packets before sending them to their destinations. This would tell destination nodes to slow down their requests for data. Another approach called the Backward Congestion Notification (BCN) has been introduced to correct congestion whenever it takes place. It happens that a destination node marks a backward congestion notification bit of

backward packets [135] to throttle data forwarding. Reactive congestion-control schemes can sometimes lead to slowing down data transfer more than should be. They can also delay the system recovery from a slow phase, especially if the traffic is changing rapidly [138].

2.7.2 Proactive congestion-control

Proactive congestion-control algorithms use information about active flows at different points of the network to determine the optimal transmission rates [139–141]. As mentioned before, a reactive control first allows congestion to take place, and then starts breaking it. However, sometimes accepting a number of packets that surpasses the network capacity results in blocking the whole structure, and deteriorates the overall performance. A preventive congestion-control necessitates early knowledge about the characteristics of the traffic. Upon having the necessary information, the network determines whether it permits the requested data or not [135]. The preventive scheme is said to be more efficient than the reactive one. A straightforward implementation of proactive control, relies on a centralized rate allocator which maintains all paths, and sends rates of all flows across the network. The central allocator – also – updates the database of flows and re-evaluates the data transmission rates. Though any centralized control is inherently unscalable, distributed implementation have been proposed [141]. The distributed proactive control performs well. It converges faster and scales to higher speeds [138]. As for packet-switches, proactive congestion-control takes preventive measures before buffers become full. A packet issues a request to reserve a buffer space on its route to its destination. Unless it is granted a place in the downstream buffer, a packet is not removed from the input queue where it is originally stocked. This approach avoids packet drops and the need for a hop-by-hop backpressure signalling [138].

In addition to congestion-control, flow control and buffering strategies have a direct impact on the memory requirements in the network, and affect the system performance.

2.7.3 Flow-control

The traffic characteristics are sensitive to load variations. In packet-switches, the buffering strategy and flow-control are tightly coupled. They both have direct impact on the memory utilization. Flow-control is essential to manage packets transfer. As depicted in Figure 2.17, the Flow-control mechanism administrates the data flow between elements of the switching architecture so that data can be handled at efficient pace [129]. As in a highway, the ultimate idea of flow-control is to keep any excess out of the network. Too much data arriving to the network can cause overflow. Unless anticipated measures are taken, data would be lost, and the system performance would severely degrade. Most

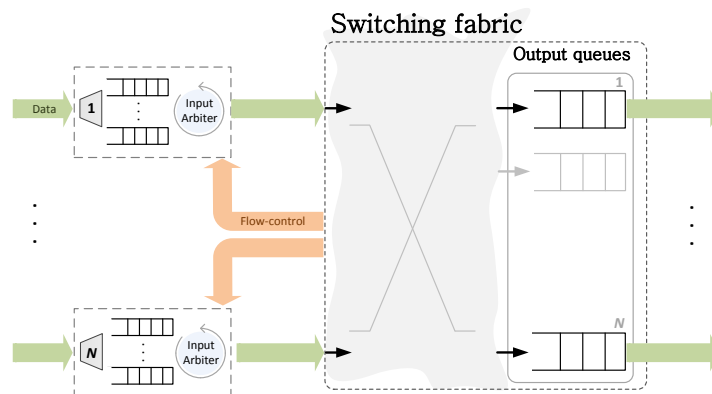


Figure 2.17: The flow control ensures that the switch is working fine and that any buffers saturation is avoided [22]. The control signal can be back forwarded from the internal buffers of the switching fabric (1bit/queue) or output buffers that are associated to the switch egresses (N bits/priority).

communication standards adopt the credit-based flow-control in ATM switches [142]. The principle is much like controlling floods where all buffers on the path help prevent congestion or packet loss by applying a flow-control process. Notifications are sent to downstream elements through a link (physical link or a virtual channel) in order to trigger data forwarding. The credit indicates the availability of a downstream buffer. It decreases at each time an upstream node sends some packets by the same number of units used in the buffering

space. Next, the performance metrics that are often used to analyse the switch performance are presented.

2.8 Performance metrics

The performance of packet-switches can be judged on many basis. In addition to the characterization of the switching topology – which is closely related to its blocking attitude – other aspects such as the routing algorithm and the buffering strategy are said to directly influence the two major performance metrics: The packet delay and throughput of the switch. The nature of the switching fabric (single or multistage network), also affects the packet latency since more stages obviously mean much longer delay. The throughput is the maximum capacity that the switch can achieve in terms of packets that are successfully conveyed to their output ports. It can be largely determined by the architecture and the scheduling process. Special interest is given to the average packet latency which helps interpret the sojourn of packets in the switch. The general term latency (with no further specifications) encompasses the delay of queuing, transmission through links, contention resolution, etc. Design goals agree on maximizing the *instantaneous* throughput and minimizing the *average* end-to-end packet latency. Other charts include the minimization of the blocking, packet loss ratio and more importantly, the hardware and software complexity. The ultimate objective is to simultaneously achieve all of the aforementioned performance levels. However, the answer to: “I want a packet-switch that is scalable, has low latency and achieves high throughput” is: “Choose two”[†].

2.9 Summary list

1- In this chapter, the buffering techniques in a context of packet-switch design are reviewed.

[†]A more general version for the appears in RFC 1925, *The Twelve Networking Truths*. “Good, Fast, Cheap: Pick any two (you can’t have all three)”.

2.9 Summary list

- 2- IQ switches are impractical because of their poor performance. If input FIFO queues are used to buffer packets at each input port, then only the first cell in each queue is eligible to be forwarded. As a result, FIFO input queues suffer the HOL blocking problem.
- 3- Many techniques have been proposed to reduce the HOL blocking. The long-standing solutions include VOQs and appropriate packet-scheduling algorithms.
- 4- The output queueing approach increases the bandwidth of the internal interconnect. It allows multiple packets to be forwarded at the same time to the same output. The main advantage of the OQ is that all packets are delayed by a fixed amount of time, making it possible to control the delay through the switch. This queueing scheme provides absolute or statistical performance guarantees.
- 5- Many other buffering schemes have been proposed to implement high-performance packet-switches with less design cost and reduced scheduling complexity. These alternative solutions include the CICQ, CIOB, buffered fabrics, PBC, etc.
- 6- In this chapter, the key defects of single-stage packet switches that disqualify them from the context of demanding DCN are stated.
- 7- The single-stage design lacks scalability, and cannot be adopted to build high-radix packet-switches given their prohibitive cost and complexity of implementation.
- 8- Multistage switching architectures are good candidate to build large scale and cost-effective switches. In this chapter, some of the well-investigated multistage interconnects that have been abundantly present in the literature, and adopted in commercialized switches/routers products are overviewed.
- 9- Networks-on-Chip have been recently adopted to make single-stage packet switches. NoCs offer a variety of interesting features such as scalability, expandability, speedup, and path diversity. While they are made to re-design the switching fabric, NoCs help resolve some of the limitations of the conventional crossbar-based switches. In this chapter, the NoC architectures as well as the packet switching modes, and the classes of on-chip routing algorithms are presented.
- 10- This chapter also provides some reading on the methods of packet re-ordering, congestion-control and flow-control mechanisms used in packet-switches.

2.10 Conclusions

High-performance packet switches/routers are vital to the performance of any network, especially to a data center network. Commonly, hierarchical switching fabrics are built to manage the floating traffic in DCNs. Single-stage crossbar switches do not meet the growing networking requirements. While they can be implemented for small-sized switches, they become quite complex to implement and unscalable to growing port counts. Multistage switches where many smaller crossbar fabrics are arranged in cascade have been a typical solution for commercial high-speed routers. They can be incrementally expanded by adding more modules to the existing design. They also have numerous benefits such as being partially or completely non-blocking, providing good broadcast and multicast features, and building in reliability with no or minimum failure in the system. While extensive work have been advocated to MSM, SMM and MMM, etc. Further proposals are yet to be studied, and more design alternatives are to be considered. This thesis is one step towards this direction.

3

A partially-buffered Clos packet-switching fabric

3.1 Introduction

Bufferless Clos-network switches are much cheaper than buffered ones, which makes them attractive. However, they reach their limitations with large port counts and high data rates. Electronic switch fabrics beyond reasonable size (e.g. 128×128) and data rates (10 Gbps) become very expensive, given the ever-increasing power and chip count implementation requirements. Moreover, centralized schedulers on which rely bufferless packet-switches need to maintain records for all ports in order to correctly assign conflict-free paths to matched sets of input/output ports. In which case, increasing the switch valency, results in high and infeasible scheduling complexity. Buffered Clos-network switches solve some issues that bufferless architectures encounter. In particular, internal buffers offer some over-provisioning of the traffic admissions, allow to implement distributed scheduling and – consequently – lead to better performance. In a classic MMM switch, CMs keep N^2 internal buffers (with N being the number of I/O links of a single CM). However, a major weakness of buffered multistage fabric architectures comes from the fact that the number of internal buffers grows quadratically with the switch size. This makes the MMM expensive, and less appealing for large-scale switches/routers. Memory is the cost-consuming component in high-speed switches. It influences both the performance and the design cost. Motivated by the pressing need for a better and efficient use of memory, the following idea is developed:

***Idea.** *“If a multistage switching architecture with few small-capacity internal buffers is built, then the packet scheduling process can be made much more efficient”.*

Single-stage Partially Buffered Crossbar (PBC) was introduced [36] to exhibit the performance of a fully buffered switch at a cost comparable to a bufferless crossbar switch. The application of partial buffering in multistage switches was discussed in [74], whereby distributed packet scheduling was described. As it will be shown later, the scheduling process in the multistage PB switch is a mixture of unbuffered and buffered crossbar scheduling. In this chapter, a three-stage Clos-network architecture with partially buffered CMs is presented. The performance of the proposed switch is investigated when one of the two following packet scheduling processes is employed: A first scheme that adopts multiple independent schedulers working in a pipelined fashion, and the second algorithm that introduces a central controller to manage in-sequence packets delivery.

3.1.1 Packet buffers in multistage switches

Buffers in IP switches/routers reduce the packet loss and absorb transient bursts of traffic. They are also an instrument to prevent output links underutilization during times of congestion. Hence, it is primordial to design packet buffers with correct sizing and placement to maximize the performance. There is no big difference between single stage packet-switches, and multistage switches in terms of packet buffering. Multistage switches are solely made of switching elements that – when isolated – can operate as single-stage modules. Packet buffers are, in general, arranged as a set of one or more FIFO queues (for switches/routers build with many service classes). They can be located on:

1. **Ingress line cards.** Queuing on the input line cards mainly depends on the traffic characteristics and the perceived performance [143]. It is common that packets arrive in bursts to ingresses of the switch, and so would they compete for service. This necessitates some buffers to

temporarily hold packets before the arbiter indicates to the memory manager which packet to serve. Commercialized switches/routers are build with a combination of small fast SRAM and large slow Dynamic RAM (DRAM) or Synchronous Dynamic RAM (SDRAM) [144].

2. **The fabric.** The common practice is to hold relatively large memories at line cards and small memories at the switching fabric. The arriving packets are more likely to contend for internal links and output ports. So packet buffering within the switching fabric can be made to increase the number of packets transmitted from the input line cards to the fabric, where they are held in small buffers awaiting for their service turn.
3. **Egress line cards.** At the output front, buffers are needed to resolve the output contention, as many packets might wish to go to the same output port simultaneously [145]. If multi-path switching fabric is used, then additional small buffers might be used to re-assemble packets and to put them back in order [90].

3.1.2 Buffers everywhere

Memoryless packet-switches have the advantage of owning a simple fabric architecture (and so, an easily scalable data plane), and in-sequence packet transfer guarantee. Yet, the scheduling process is highly complex, which results in an unscalable control plane. Furthermore, bufferless switches yield poor performance under unbalanced traffic patterns. To shorten the switch configuration time and to ameliorate the system performance, buffered switching elements were added to several stages of the multistage switching architectures. An MMM packet-switch is a straightforward alternative to scale up the single-stage buffered crossbar switch. It absorbs all contentions, and offers backlog buffers at all stages to temporarily store unscheduled packets [77]. In a three-stage Clos-network with non buffered middle stage modules, packets reach the egress line cards orderly. However, in-sequence packets delivery matters in buffered multistage switches, as packets are more likely to experience variable queuing delays in the internal buffers. In [75], authors described the MMM^e which implements per-output flow

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

queues at the middle stage of the switch. The *TrueWay* switch was proposed in [86]. It is a multi-plane MMM that provides in-order packets delivery using the hashing technique to allocate a path to each flow of packets. A more recent work [12] discussed an MMM switch with batch scheduling for in-order packets delivery. Although they simplify the scheduling process in multistage packet-switches, fully buffered architectures are prohibitively costly.

3.1.3 Few internal buffers

A clearly preferable alternative to the previously described switching architectures, is a partially buffered Clos packet-switch. This solution comes in between the bufferless and fully buffered design. The scheduling process can also be inspired of both, a centralized and a distributed approaches. Chrysos. *et al* discussed and evaluated an MMM switch where all central switching elements are fitted with buffers [74]. Instead of a dedicated buffer per crosspoint, a shared memory per output link was considered, and multiple pipelined arbiters were used to schedule resources. Recent works [138] proposed a proactive congestion control scheme (to their previous work in [74]) for better QoS.



Some background: Single-stage partially buffered switch fabric

Originally, multiple crossbars were interlinked to make multistage packet-switching architectures. Diverse combinations of crossbars and internally buffered fabric were later introduced to meet the design goals and the pre-set hardware and software requirements. The bufferless switching modules although simple and cheap, impose quite complex scheduling algorithms to resolve the input and output ports contention. Meanwhile, internally buffered fabrics relax the scheduling algorithm complexity by distributing the arbitration process over many sub-schedulers [146]. All the same, the fabric involves as many internal buffers as the number of crosspoints (see Figure 3.1), which limits the scalability of such an architecture and restricts its application to small-scale switches [19].

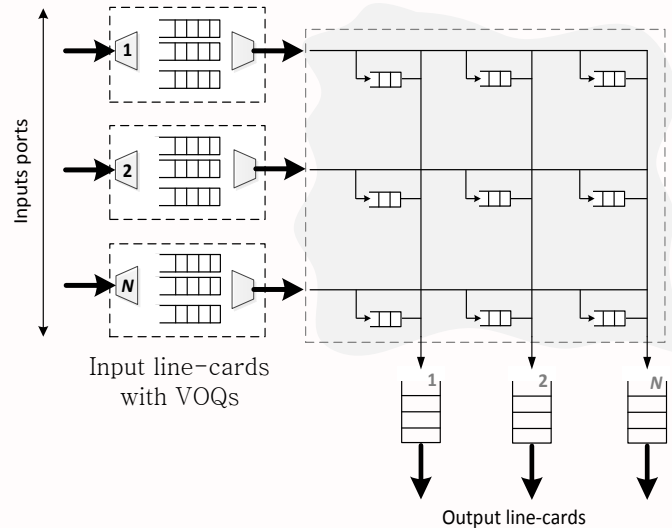


Figure 3.1: $N \times N$ buffered crossbar switch with N^2 crosspoint buffers.

In [36], the PBC switch was introduced. The idea is about using a mixture of design aspects that inherently belong to bufferless and internally buffered crossbar switches. In contrast to fully buffered fabrics, the PBC switching fabric encloses few number of *physically separate* queues per output, as illustrated in Figure 3.2. To keep the bandwidth requirements low, NB internal buses ($B \ll N$ and B buses are associated to each output port) are used. Buses are assumed to run at the same bandwidth as the external line rate. The scheduling process in the PBC switch relies on distributed arbiters. Input Schedulers (ISs) manage packets transfer from input line cards to the fabric buffers. In coordination with the Grant schedulers (GSs), ISs select an eligible VOQ to serve in the next time-slot. The GSs monitor the internal buffers. They keep records for the available buffer space per output port and communicate with ISs through the grant queues. The last scheduling stage concerns with removing packets from the PBC fabric and forwarding them to their corresponding output ports. To this end, Output Schedulers (OSs) arbitrate packets departure from the internal buffers adopting a specific selection logic (e.g., Oldest Cell First (OCF) policy to preserve the packets order). The Distributed Round Robin (DRR), DROP and Prioritized DROP (DROP-PR), is a set of pipelined scheduling algorithms that were suggested along with the PBC switch [71] to fulfil a wide range of performance needs.

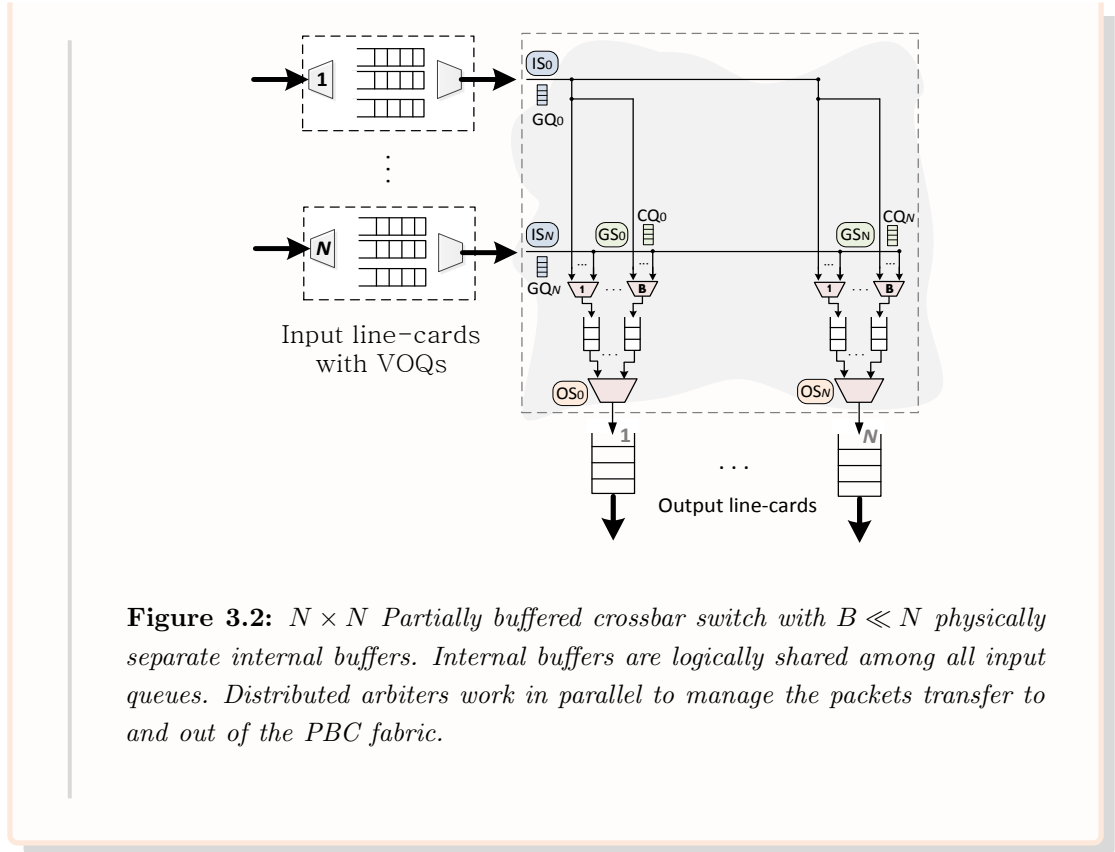


Figure 3.2: $N \times N$ Partially buffered crossbar switch with $B \ll N$ physically separate internal buffers. Internal buffers are logically shared among all input queues. Distributed arbiters work in parallel to manage the packets transfer to and out of the PBClos fabric.

In the following section, details of the switching architecture and some design challenges are given. In subsequent parts, the packet scheduling process is provided. Next, the performance of the PBClos switch is assessed under different traffic types.

3.2 Architectural design challenges

To build a scalable and cost-effective multistage packet-switch one has to consider two aspects of major impact on the performance of the switch: The data plane and the control plane. While bufferless multistage switches exhibit a scalable data plane (as a consequence of the memoryless switching fabric and the absence of any need for packet resequencing mechanisms), their control plane (that reports to the centralized arbiter that sets up resources reservation) imposes a very heavy toll on the system. To this end, a centralized scheduler has not been reported to scale beyond 128 ports [112]. For buffered packet-switches, it is the

counter case. The multiple internal buffers (usually shared memory type) are a big handicap. Meanwhile, the packet scheduling process, often leans on many separate sub-arbiters operating in a pipeline, and so it is liable to scale to large valencies. The perpetual challenge in packet-switch design, is about how good and efficient is, a particular switching architecture, or the other?

In this chapter, concepts inherited from bufferless and fully buffered multi-stage packet-switches are merged. Using a limited number of crosspoint buffers at the central Clos-network stage, and appropriate scheduling algorithms, a practical switching fabric architecture is described, and the switch performance is evaluated under a range of traffic patterns.

3.2.1 Over-provisioning in conventional MMM switches

Conventional MMM switches, adopt fully buffered modules in the central stage of the Clos-network. They use dedicated crosspoint buffers in CMs [12], or shared memories to serve as a buffer for a set of output modules [138]. The capacity of the internal buffers may vary from one packet to several packets. Experiments have shown that scaling up a crosspoint buffer size to several packets, boosts the switch performance, especially if the traffic arrivals are irregular or bursty [74, 77]. Provided there is enough space, packets often find a room in the middle stage crossbars where they wait to exit CMs to the output stage of the switch instead of being discarded [12]. Note that, holding many internal buffers in the middle-stage modules or/and increasing their capacities, is carried out to cater for the amount of packets dispatched at the input scheduling phase. However, both measures are without interest in the output scheduling phase. Actually, if no speedup is used, output schedulers serve one packet from an internal buffer among those dedicated to an LC link, at each time-slot.

The rule of thumb says that the more intermediate buffering is available, the less a central control is needed. That is why in the bufferless MSM switch [64], a centralized scheduler must be able to find a conflict-free matching between a large set of requests and output links. The goal is to contribute towards two major aspects:

- A less expensive switching architecture.

- A good scheduling that provides high performance.

In this chapter, only few crosspoint buffers are to be implemented (much less than in common MMM switches). An internal buffer can store only one packet. In addition to the distributed approach, a frame-based packet scheduling is presented. The frame-based scheme simplifies the central scheduler complexity as compared to the MSM switch as it will be shown later.

3.2.2 In-order packets forwarding

In-order packet delivery matters in multistage switches. The three-stage Clos interconnect provides multiple paths to route packets from any input stage module to any output stage module. The presence of buffers at the central stage of the Clos-network switching architecture potentially leads to an out-of-order packet transfer. In general, cells are scheduled to cross different CMs. They are likely to experience variable queuing delays in the buffered fabric. Therefore, they reach to their OMs disorganized. New design trends rise many challenges and questions: Can a performance comparable to that of fully buffered Clos switches be provided using only few internal buffers at the CMs? Is it possible to forward packets to their output ports without order disruption and with no performance degradation? Can all of the aforementioned objectives be achieved with a simple scheduling scheme? These issues are tackled, and a possible switch design that meets all these objectives combined is discussed. In what follows, the PBClos switching architecture, the packet buffers, and the scheduling mechanism are described.

3.3 PBClos: High-level switching architecture terminology

A high-level abstraction of the switch architecture is given in Figure 3.3. Ingress Line Cards (ILCs) do the variable-size packets segmentation into fixed-size cells before they enter the IMs. Egress Line Cards (ELCs) reassemble received cells into packets, and finally, send them out of the PBClos switch. A three-stage

3.3 PBClos: High-level switching architecture terminology

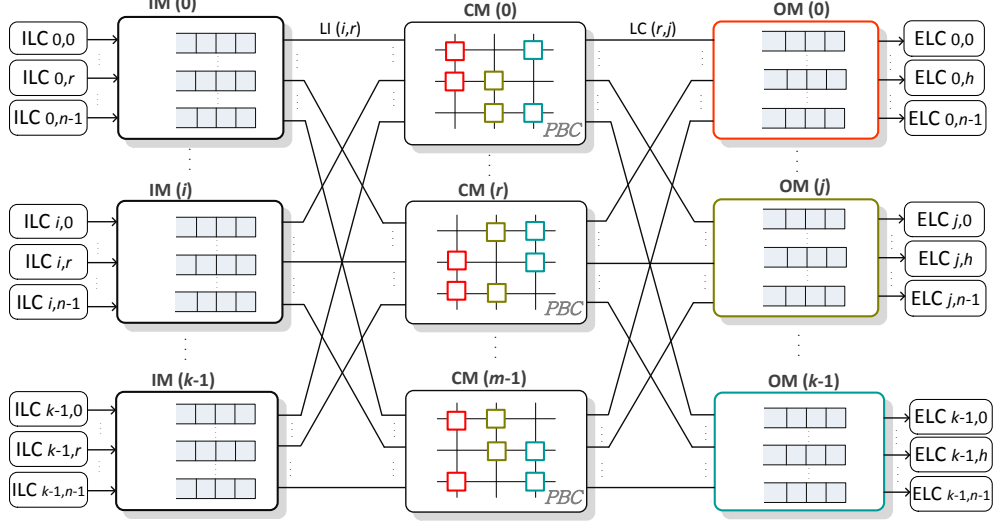


Figure 3.3: $N \times N$ three-stage PBClos switch architecture.

Clos-network switch made of buffered modules is described. The switch operates on fixed sized cells where packets of variable length get segmented into fixed size cells while inside the switch, and are reassembled back to packets upon their exit. There are k IMs, each has a dimension $n \times m$. IMs are connected to the central modules by means of m links $LI(i, r)$ (where i is the i^{th} IM and r is the r^{th} CM; $0 \leq i \leq k - 1$ and $0 \leq r \leq m - 1$). The third stage consists of k OMs, each of which is of size $m \times n$. Buffers at the input stage are organized into N VOQs to eliminate the HoL blocking ($N = n.k$). Every queue stores cells coming from an input port in $IM(i)$ to an output port $OP(j, h)$. CM switches are partially buffered crossbars [36]. A $CM(r)$ has k output links, each of which is denoted as $LC(r, j)$ that connects it to $OM(j)$. An $OM(j)$ has n OPs to which is associated an output buffer. An output buffer can receive at most m packets and forwards one packet to the output line at every time-slot. Unlike traditional crosspoint queued crossbars, a PBC crossbar contains only a small number of internal buffers; $B \ll k$. A total of B separate internal buffers are maintained per output link, LC. For simplicity, the Clos-network's expansion factor is set to $\frac{m}{n} = 1$, which makes the architecture a *Benes* network; the

lowest-cost practical non-blocking fabric. However, all the descriptions remain valid for any non-blocking Clos-network settings.

3.3.1 Distributed scheduling in the PBClos switch

The process of packet scheduling can be either centralized or distributed. In some switching architectures, both approaches might be used to trade-off complexity to performance. In this section two different schemes to schedule packets in the PBClos switch are introduced. Both methods are proposed to achieve specific goals, in addition to high performance.

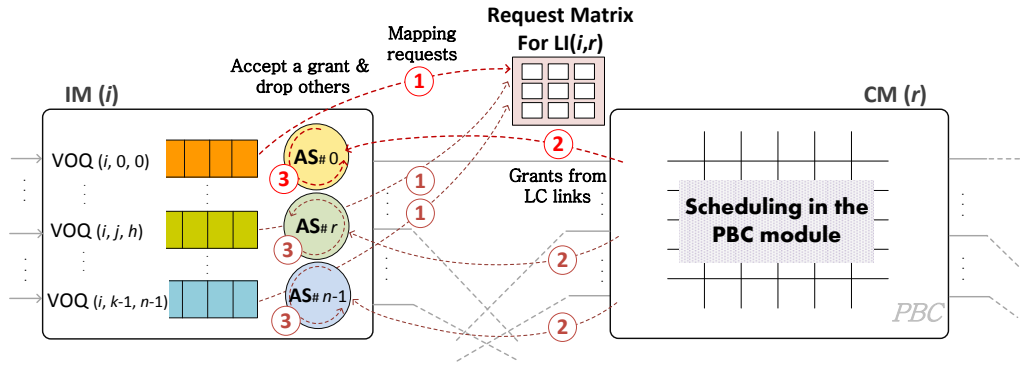


Figure 3.4: PBClos switch with distributed scheduling. Grants are forwarded from CMs to active VOQs. A VOQ that receives more than one grant from a CM (i.e., The CM has room to house a packet to the corresponding OP), accepts one grant in a RR fashion.

Distributed arbiters avoid the scalability limitation of a centralized approach by dispersing the scheduling function over the switching modules of the Clos-network. First, a distributed packet scheduling is proposed, as it is more straightforward in the context of buffered multistage switches. A total of n Accept Schedulers (ASs) in every IM are implemented (one per input port) and k Output Schedulers (OSs) are used in each CM (one per LC link). Independent arbiters operate in pipeline to help spread the scheduling functions over multiple control chips, and to run at high speeds. Figure 3.4 depicts the succession of events over time. During the packets injection, up to n new packets arrive to an IM and get stored to their corresponding input queues. The path allocation is

3.3 PBClos: High-level switching architecture terminology

made in the upstream direction, starting from the second stage of the switching network to the first stage. An AS chooses the first non-empty VOQ that appears in its RR selector and marks it as active (or also, eligible for scheduling). Clearly, the distributed packet scheduling scheme, provides no visibility on the status of the internal buffers located at the middle stage fabric. Hence, an active VOQ sends requests to all CMs.

At the starting of the scheduling cycle, the source/destination tuple of the HoL packets are copied to the *Request Matrix* of a LI link (event ① from active VOQs to the *Request Matrix*). A *Request Matrix* that is associated to $LI(i, r)$ serves to map all active requests that come to a $CM(r)$ through $LI(i, r)$. It can be implemented using simple counters. The mapping action does not include a real cells transfer. It just sends a request signal to trigger the PBC modules' schedulers in the next pipeline stage of the scheduling. Output schedulers send grants to requests (event ②, from the CMs to VOQs) and ultimately, accept schedulers select, each, one grant (event ③).

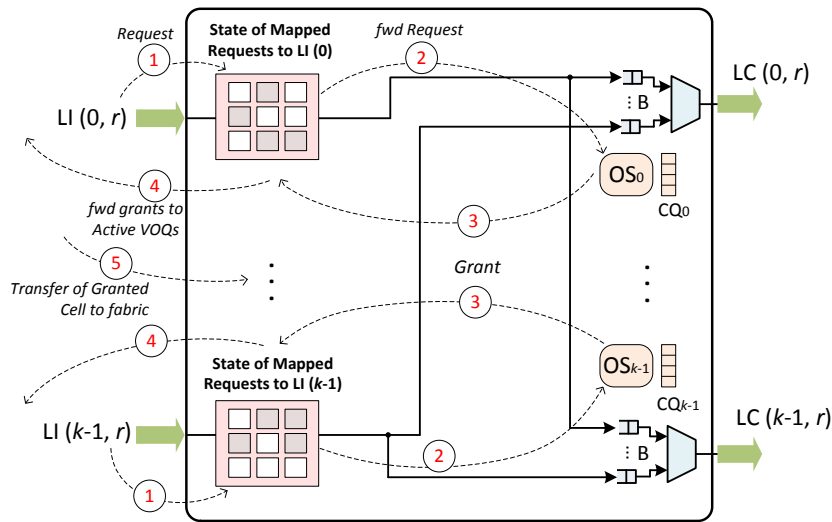


Figure 3.5: Scheduling in the central stage PBC switches.

3.3.2 Grant and accept scheduling

The internal buffers run at the external line rate. Isolated buffers are used in order to make sure that a low bandwidth is maintained, and to avoid the need for high-throughput shared memories as those described in [74] and [36]. Figure 3.5 shows the grant mechanism in a central module. Each internal buffer can house at maximum one packet. Output schedulers hold Credit Queues (*CQs*) of size B , to record the number of empty internal buffers per LC link. Credits are decremented at each time a grant is sent to a mapped request in the *Request Matrix* (event ③ in Figure 3.5) and they get replenished at the end of the scheduling cycle; when packets exit the fabric. At the end of the grant phase, grants are directly forwarded to active VOQs in the IMs (event ④). If an active VOQ(i, j) receives more than one grant from different CMs, the associated AS(i) accepts the one that appears in the RR priority and drops unaccepted grants. Next, the packet is transferred to its corresponding CM (event ⑤).

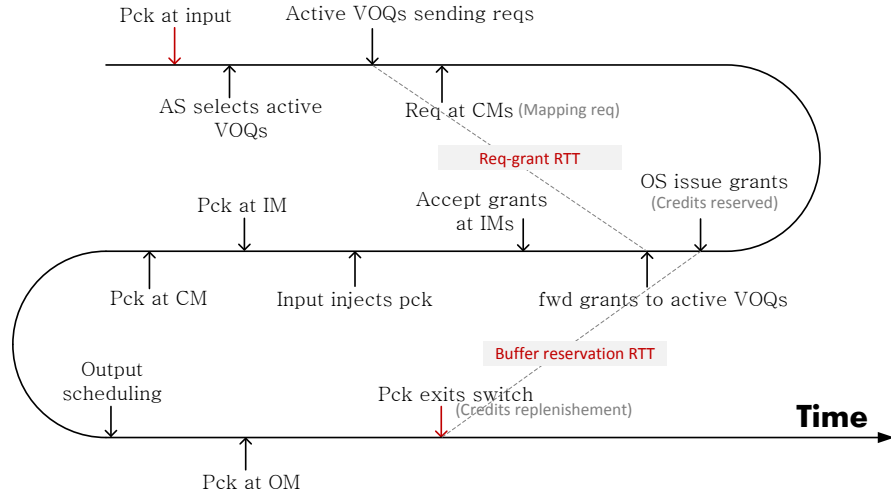


Figure 3.6: Pipelined scheduling in the multistage PBClos switch using distributed schedulers. Internal buffers are managed using a credit-based flow-control. The req-grant Round Trip Time (RTT) starts from the time requests are sent to CMs, until grants are received at input ports. Buffers reservation RTT spans the instant credits are reserved in the CM switching fabric, until they are released when packets finally exit the switch.

3.3 PBClos: High-level switching architecture terminology

Note that, a matched LI link is marked as reserved to avoid clashes. It automatically gets excluded from the arbitration set of the remaining active VOQs in other IMs. The action of mapping requests, the grant phase and the output arbitration are all independent. They can be pipelined and run in a parallel way as shown in Figure 3.6. While a scheduling cycle ends, and cells leave the PBC modules, the buffers reservation of freshly mapped requests takes over. Packets that arrive to the output stage of the Clos-network find their ways to their OPs.



Practical considerations for distributed scheduling

The excellent performance of buffered multistage switches* with distributed schedulers has been proven analytically [147, 148] and by simulations [74, 112, 149]. This was always subject to one condition that a distributed scheduling approach is likely to guarantee: No output port is overwhelmed with traffic loads [148]. Most often, packet switches are described in the literature to operate in discrete phases (arrival, transfer, and departure). This assumption, although it simplifies the system description, is not always valid in the real world. Systems can be build to adhere to this assumption. However, this would require a period during which data transfer is suspended until the scheduling process ends. Pipelining is the technique to be implemented in order to cope with this insufficiency. Traffic is received at first place. During the update period, it is handled to start the current scheduling process. In the next update period, traffic is forwarded to the corresponding outputs. In practice, this would cause packets extra delays of one or two update periods that can be all concentrated at the ingress ports using an hierarchical scheduling [149]. Delays still happen when the traffic rate suddenly increases at the input ports. It is possible to allow a more continuous implementation with ports sending their status information and asynchronously transferring and updating rates on the VOQs. A distributed scheduling has an important limitation which is the overhead caused by exchanged messages. Practical variants choose to schedule data packets which size is larger than that of the exchanged messages [150].

*For buffers that can hold a small number of packets; typically between 2 and 16 packets, each, as commonly described in the literature. The important things, is that the switching fabric must have sufficient internal storage capacity to temporary handle any short-term congestion that may happen during an update period.

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

Although practical and – relatively – simple to implement, the distributed control ends up routing packets of the same flow across several CMs. Packets remain in the internal buffers until it is their turn to exit the fabric (output scheduling phase). This obviously results in out-of-order packets delivery. This problem is subtly alleviated instead of allowing packets to reach their output ports out of order and then implementing complex resequencing mechanisms.

***Idea.** “If packets of the same flow are forwarded throughout the same path internally, there would be no need for a re-sequencing logic within a flow.”

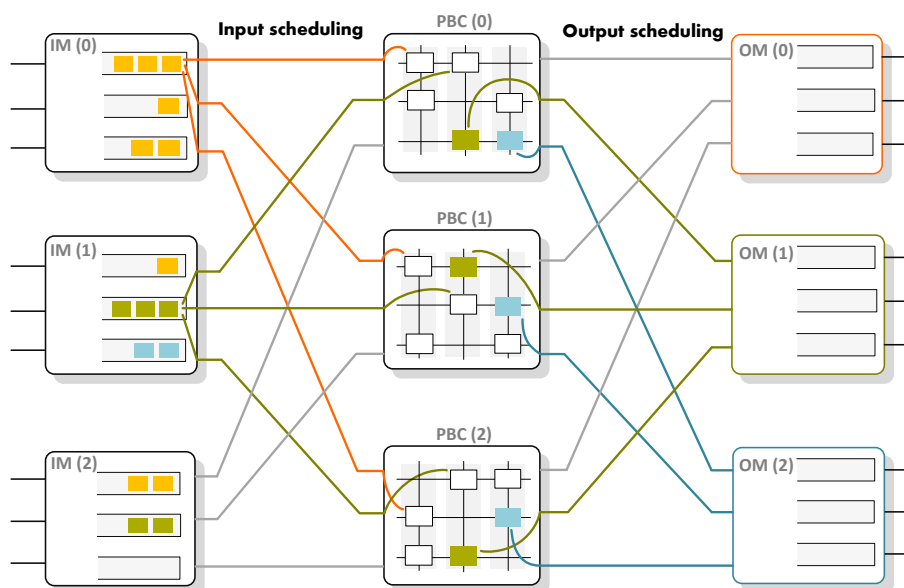


Figure 3.7: An example of batch scheduling in the PBClos.

3.4 Centralized packet scheduling

In the context of buffered switching architectures, a distributed packet scheduling is a straightforward option. The scheduling phases might be pipelined for faster execution. More importantly, the distributed approach do not bound the

scalability of the switch as a dispersed arbitration necessarily decouples the data plane from the control plane, and proves more effective for high-radix, high-performance packet-switch design. All the same the distributed scheduling do not assure in-order packets delivery. Instead of adding re-ordering buffers to the first or last stage of the Clos-network [77, 86], the organization of the input buffers in the IMs is altered, and a frame scheduling rather than a packet-per-packet scheduling mechanism is proposed. The resulting architecture differs from the single stage PBC and from the PBClos with distributed arbiters. As it will be described later, the organization of the input buffers at the IMs is totally different from other proposals. A central controller is implemented and a frame-based scheduling is used to transfer packets from a stage to the other.

IMs are organized into Virtual Output Module Queues (VOMQs), where packets are enqueued according to their destination OMs. VOMQs are shared memories that can receive packets from up to n input ports at a time, and send up to m packets to different CMs. The choice for VOMQ queueing is done in accordance with the frame scheduling itself. The idea is the following: A flow of packets is always transferred through the same path to avoid any resequencing logic within the flow. The approach is similar to hash-based routing solutions, where several hash functions are used to assign routes to the different packet flows. However, the ultimate purposes of the two schemes are different (load-balancing for the hash-based routing versus ordered packets delivery for the frame scheduling). The arbitration in the PBClos switch is made of the following two phases: The input scheduling which consists on sending packets from the VOMQs to the internal buffers in the PBC modules, and the output scheduling that concerns with forwarding packets from CMs to their final output line cards.

3.4.1 Input scheduling

Figure 3.7 gives an example for the frame scheduling in a PBClos switch. In a VOMQ, m consecutive packets would make a *frame*. Unlike previous proposals [12, 74, 75], where dedicated crosspoint buffers with variable capacity are available, the current switch design imposes several constraints: In a PBC

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

module, the number of internal buffers per LC link (*i.e.*, OMs) is very limited. A buffer can accommodate only *one* packet. Besides, the buffering space is logically shared between all LI links (*i.e.*, IMs), and – hence – it needs a pertinent control scheme. A central controller is used to track the buffers availability in all CMs, as depicted in Figure 3.8. For every input module, the controller marks as eligible, all VOMQs with full *frames* for which there is enough space in the PBC fabric. Ties are broken in a RR fashion and only one input queue is elected at a time. The selected VOMQ broadcasts packets to CMs. The status of the queues at the input modules is communicated to the central arbiter using a $k \log_2(m)$ wide signal. The availability of the internal buffering space at a middle-stage module is sent on a $k \log_2(B)$ wide signal, from the CM to the controller. The arbiter, in its turn, alerts the IM through a $\log_2(k)$ wide signal, and indicates the succession of selected VOMQs. It also sends a $\log_2(k)$ wide signal to the CM to point out indexes of outputs having available room and ready to receive packets from matched input queues.

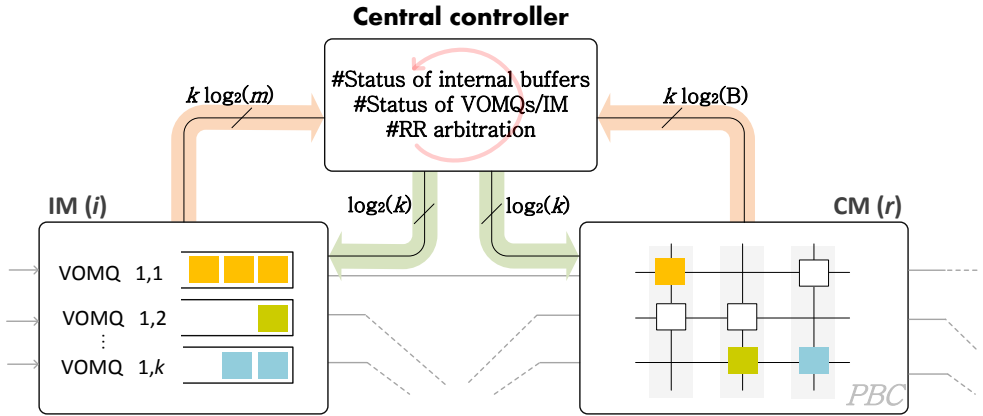


Figure 3.8: Centralized control in PBClos packet scheduling.

3.4.2 Output scheduling

In [12], authors described a MMM switch with fully buffered CMs, dedicated Crosspoint Buffers (CBs) and a centralized batch scheduling. In addition to the VOMQs selection, the scheduler processes the output scheduling. It chooses k

non-empty CB one from every set of internal buffers of an output link LC. The scheduler applies the same configuration to all CMs and packets are dequeued from the CBs in the different CMs. In this way, a batch is reconstructed again at the corresponding OM. As for the PBClos switch, *physically* separate CBs are used to keep the memory requirements low. Subsequently, the output schedulers can be embedded and more compact CMs are obtained. The set of internal buffers that belong to a given LC is called a column of CBs. The CBs might be located anywhere in the column since the access to the buffering space is *logically* shared between all IMs. Moreover, no restriction on how to locate the CBs in the different CMs is imposed. Instead of entrusting the output scheduling to the central controller, simple and distributed output schedulers are used as it is described in Sub-section 3.3.1. Since packets of the same batch arrive to internal buffers at the same time-slot, serving packets in a First-Come-First-Served (FCFS) manner preserves the packets' sequencing. Next, the PBClos switch's performance is evaluated under many traffic profiles. The performance of the switch is also compared to the MSM switch and variants of the MMM switch.

3.5 Performance analysis

A rough comparison between the PBClos switching architecture, the MSM switch and two variants of MMM switch is given. In TABLE 3.1, are stated some of the HW requirements of the different switching architectures. Note that the first difference has to do with the internal buffering space. The PBClos switching fabric maintains buffers per LC link unlike the MMM variants that use a dedicated crosspoint queue for every output link. The PBClos switch contributes towards reducing the total number of crosspoint buffers from k^2 to $(k \cdot B, B \ll k)$. The second architectural difference to be mentioned, is about the type of the internal memory implemented in each switching fabric. In [138], authors describe a shared memory. However, both buffered switch presented in [12], and the PBClos switch call for physically separate queues. At the scheduling front, the MSM switch, the current proposal with a centralized control, and the MMM switch as described in [138] guarantee in-order packet delivery. The central arbiter of the MSM switch needs cater for $m \cdot N$ queues, for

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

which it is necessary to manage and coordinate the matching. Yet, the PBClos and MMM [12] switches greatly reduce the scheduling complexity and the scheduler size from $\mathcal{O}(m \cdot N)$ to only $\mathcal{O}(\sqrt{k})$. A step further in the assessment of

Table 3.1: Comparison between the different switching architectures.

	MSM [64]	MMM [12]	MMM [138]	PBClos
Type of the fabric	bufferless	buffered	buffered	partially-buffered
Type of the scheduling	centralized	centralized	distributed	distributed (3.3.1) centralized (3.4)
Number of crosspoint buffers	NA	k^2	k^2	$k \cdot B^\dagger$
Size of crosspoint buffer	NA	1 packet	12 packets	1 packet
Type of internal memory	NA	separate memory banks	shared memory	separate memory banks
Size of the centralized scheduler	$\mathcal{O}(m \cdot N)^\ddagger$	$\mathcal{O}(\sqrt{k})$	NA	$\mathcal{O}(\sqrt{k})^*$
In-sequence packets delivery	yes	yes	no	yes $^\diamond$

\dagger . ($B \ll k$)

\ddagger . $N = n \cdot k$

*. The explanation is the same as done in reference [12].

\diamond . If centralized frame-based scheduling is used.

the switches' performance is made and the simulation results are presented. The performance of the PBClos packet-switching fabric architecture using multiple distributed schedulers and a centralized arbiter is compared to that of the MSM switch and three variants of the MMM architecture as have been described in [12], [77] and [75]. The switch size is 256, if not explicitly mentioned and the simulation time is 10^6 time slots. The PBClos switch is interesting in the way it provides a higher degrees of design freedom. Varying the number of internal buffers helps monitoring the cost, complexity and performance of the switch. Considering full number of crosspoint buffers in the CMs and two different scheduling schemes, maps the switch to two previous proposals: Three-Stage Buffered Clos-network Switch (TSBCS) [12] (if the frame-based scheduling is used) and the Scheduled Fabric (SF) [74, 138] with crosspoints size $b = 1$ (if distributed scheduling is adopted). The notation PBClos-centr is used for the PBClos switch with a frame-based scheduling and PBClos-dist is reserved for

3.5 Performance analysis

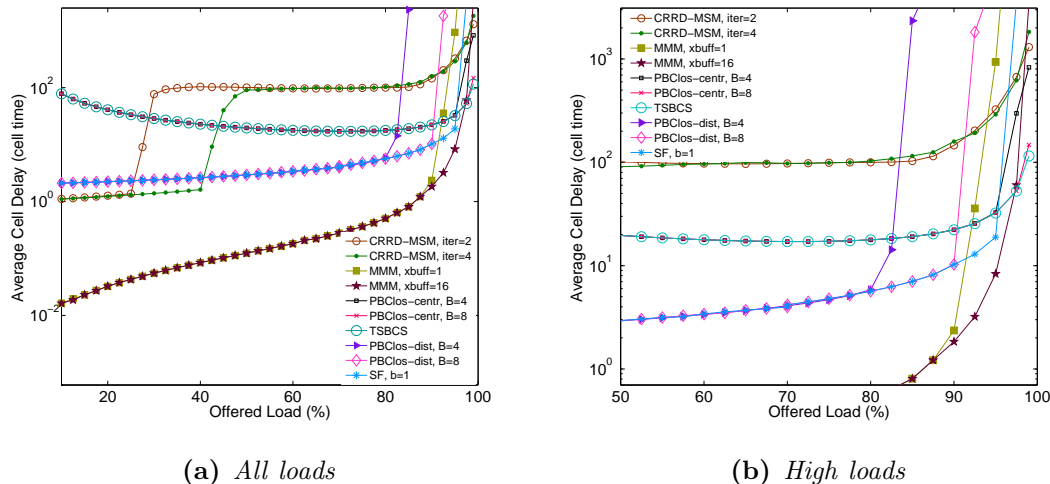


Figure 3.9: Delay performance for a 256×256 switch under Bernoulli uniform traffic.

the distributed scheduling scheme. The MMM is the switching architecture suggested in [75], and xbuff is the size of its crosspoint buffers. Details about the traffic types that are used along the thesis are provided in Appendix A. The first set of simulations is performed for uniform traffic: Bernoulli arrivals with evenly distributed destinations and uniform arrivals of bursts with a default burst size of 10 packets. The bursty traffic can be modelled as an ON/OFF process. The burst of packets that comes to a switch input port during the ON period is in destination to the same output port. Destinations among bursts are uniformly distributed among output ports. The delay performance presents the sojourn time of packets through the switch fabric. It is estimated by averaging over all queuing delays measured during the simulation. Figure 3.9(a) shows results for Bernoulli uniform arrivals. The PBClos switch has a delay performance that is in between what MSM and MMM provide. The switch performs poorly under light loads, but deals better with medium and high loads. With a distributed scheduling, the switch offers lower latency than when a frame-based scheduling scheme is used. The reason is that the VOMQs need wait for full frames to form before they are eligible for scheduling. Figure 3.9(b) is a clearer view of the delay-throughput curves shown in Figure 3.9(a). Note that increasing the number of crosspoint buffers contributes towards a better

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

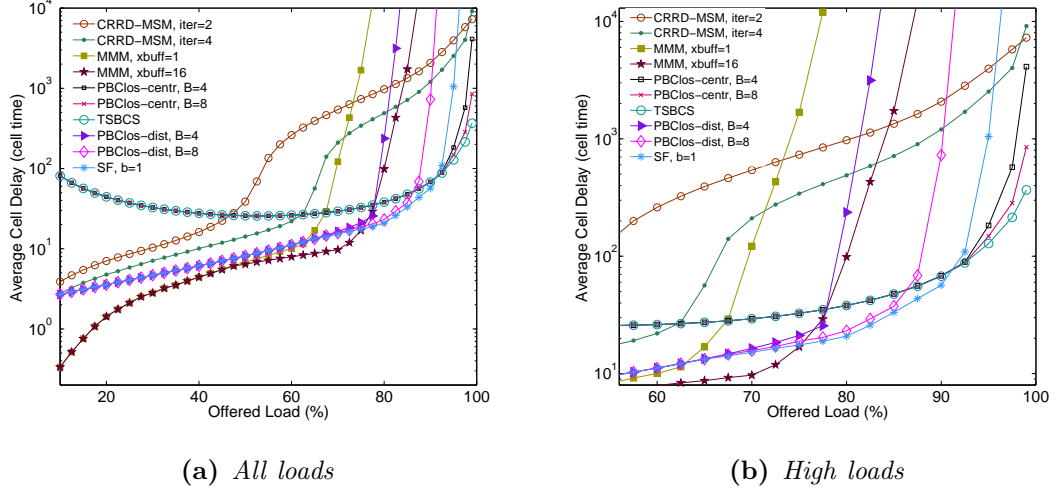


Figure 3.10: Delay performance for a 256×256 switch under Bursty uniform traffic.

throughput. For 256×256 switch, setting B to $B = k/2 = 8$ queues/LC link and using distributed arbiters, makes the PBClos achieve up to 90% throughput versus 95% for the SF. Remarkably, half of the buffering amount used in the middle stage of the SF architecture is saved for comparable performance. Using the same number of internal buffers per output link in CMs and a frame-based scheduling, PBClos seems to provide full throughput and slightly higher delay than the TSBCS switch. Figure 3.10(a) shows that under uniformly destined bursty traffic, the PBClos outperforms the MSM switch and the classic MMM switch that is described in [75]. Note that a frame-based scheduling with as few as 4 internal buffers per LC link achieves full throughput as shown in Figure 3.10(b). On the whole, the variation of the average latency of the PBClos switch is smooth. It is not affected by the switch valency while that of the MSM switch is much influenced due to its instability. The uniform arrival patterns do not reflect a real traffic. They just help trace the switching architecture’s response assuming the best case scenario. For the next set of evaluations, non-uniform traffic is considered: Hot-spot, unbalanced [36] and diagonal traffic (where input i of the switch sends traffic only to output of the same index).

Figure 3.11(a) depicts the latency of switches under hot-spot traffic. It is clear that the bufferless architecture do not adopt well with non-uniform traffic.

3.5 Performance analysis

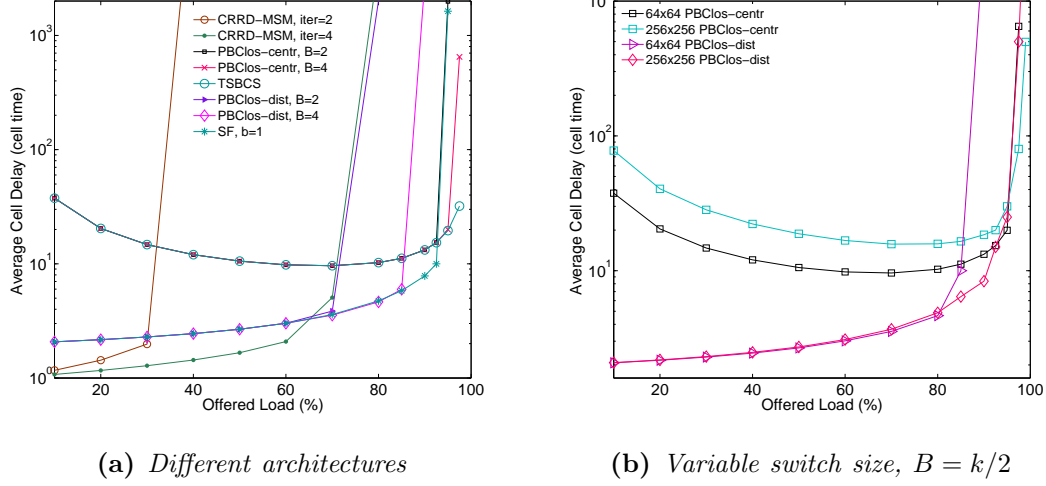


Figure 3.11: Delay performance of a 64×64 switch under hot-spot traffic.

Actually, the Concurrent Round Robin Dispatching algorithm (CRRD) with 4 iterations achieves as low throughput as 70%. In contrast, simulation results show that holding as few as $k/2$ buffers/LC in the central-stage modules of the PBClos switch, still provides high throughput performance. In Figure 3.11(b), the performance of the PBClos is assessed by varying the switch valency under hot-spot packet arrivals. Unlike a distributed scheduling, the frame-based scheme is not as efficient under light to medium loads as it is under heavy loads. It results in an initial waiting time that is proportional to the switch size (the central scheduler waits for the construction of larger frames). The distributed scheduling is insensitive to the switch size, but its throughput saturates at around 90% for a 64×64 switch and 97% for a 256-ports switch. In Figure 3.12(a) and 3.12(b), the throughput of switches under unbalanced and diagonal traffic – respectively – is given. The throughput of the PBClos switch increases as the number of crosspoint buffers in the central modules is increased. Also, the throughput is marginally affected when a distributed packets scheduling is adopted. Under diagonal traffic, both scheduling approaches make the switch achieve equally high throughput levels even when $B = k/2$.

3. A PARTIALLY-BUFFERED CLOS PACKET-SWITCHING FABRIC

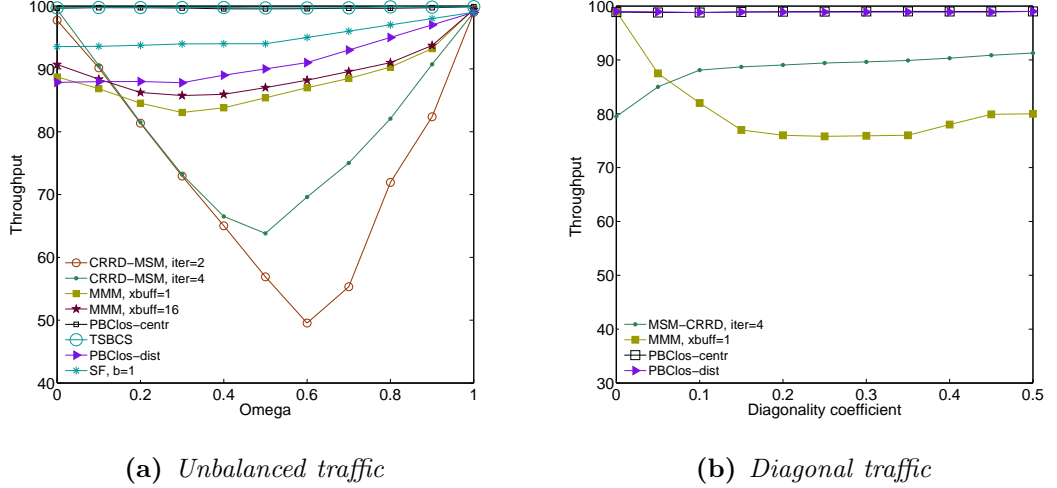


Figure 3.12: Throughput stability of 256×256 switch under non uniform traffic, $B = k/2$.

3.6 Related work

Well-studied types of multistage Clos-network packet switches involve the MSM switch [64, 69, 71]. Broadly speaking, the MSM switch (and other similar bufferless Clos-network switches) requires a two-step scheduling process before proceeding to the packets transfer. In essence, bufferless multistage switches are appealing for their low-cost. However, they use a centralized scheduler to resolve contentions, which proves to be scalability limited. It also contributes to rising the cost/complexity ratios (since they require two iSLIP-style exact matchings to be found, some of which among N ports, per time-slot) [36]. Buffered switches [12, 74, 75, 77, 86, 138, 149] reduce the complexity of the packet scheduling process. The PBClos switch is a buffered Clos-network switch with as few crosspoint buffers as $k \cdot B$ ($B \ll k$) per CM. This is to be compared to k^2 queues per CM in the conventional MMM switch. The packet scheduling mechanism is inspired from the centralized (in bufferless) and the distributed schemes (in buffered switches). This work has few similarities with the work in [36, 74, 138]. However, the PBClos switch differs in the following ways. First, the objective is to design large-capacity multistage switching architectures that fit for the demanding DC networks unlike the proposal in [36] that discussed

and evaluated a single-stage PB crossbar switch. In [74, 138], authors presented different scheduling algorithms. Second, the switches exhibit other differences that report to the crosspoint memory type and the capacity of the embedded queues. In summary, this work is a practical step toward building scalable high-capacity packet switches, where the hardware and scheduling requirements can be orders of magnitude lower than the previous proposals.

3.7 Summary list

- 1- In fully buffered switching architectures, increasing the number or the capacity of the internal buffers, is made to cater for the amount of packets dispatched at the input scheduling phase.
- 2- However, both measures have small impact on the output scheduling phase.
- 3- Partially buffered switches have been designed with emphasis on the following rule: Employ less resources, and substantially plan them!
- 4- In this context, a practical and cost-effective multistage switching architecture (PBClos) is proposed. The switch relies on partially-buffered crossbars with capacity-limited queues.
- 5- The PBClos switch comes in between a bufferless and a fully-buffered three-stage Clos switching architectures, and takes the best of both designs.
- 6- A distributed packet scheduling was proposed for the PBClos switch, whereby simple and distributed arbiters are run in a pipelined fashion to manage the packets transfer. In spite of its good performance, the distributed packet scheduling scheme does not guarantee an ordered packets delivery.
- 7- A frame-based scheduling is also proposed. This algorithm distributes packets of a flow over all CMs of the Clos-network and builds back a frame at the OMs avoiding the need for costly re-ordering buffers.
- 8- This discipline cumulates the delay when the switch is light-loaded. However it provides good performance, and – almost – constant delay variation under heavy loads.

3.8 Conclusions

The PBClos switch relies on partially-buffered crossbars with capacity-limited queues. The design comes in between the bufferless and fully-buffered architectures, and takes the best of both. The performance of the switch is evaluated using multiple, independent single-resource schedulers that work in a pipeline. In spite of its good performance, the distributed scheme does not guarantee an ordered packets delivery. Thus, a frame-based scheduling is proposed. The scheme distributes packets of a flow over all CMs of the Clos-network fabric, and builds back a frame at the OMs, avoiding the need for costly resequencing buffers. This discipline cumulates the delay when the switch is light-loaded. However it gives good performance and shows scalability to heavy loads.

4

A packet-switching architecture with uni-directional NoC fabric

4.1 Introduction

Getting new answers to the same question is challenging. In the area of packet-switching design, building reliable large-scale switches/routers has been always a hot topic. The need for high-performance switches has risen with large networks becoming more bandwidth hungry. DCNs are an instance of demanding networks, where conventionally deployed commodity switches/routers are limited in terms of scalability and poor in terms of performance. Historically, packet-switching has gone through decades of design and optimization. While switching architectures are too much diversified, the objective has remained the same: High-performance switches that scale well with any type of variation (traffic, valency, etc.), and that can be built with the minimum cost. Back to the late 70's, works on communications have made lots of theoretical and experimental achievements, mainly for packet-switches. Single-stage modules of different architectures have been extensively studied, manufactured, and distributed for commercial use. Later on, multistage networks have been introduced to inspire large and practical switching fabrics. They are made up of many single-hop crossbars and memory blocks. Recently, switching fabrics were designed with reference to the NoC paradigm. The approach offers several advantages over the classical crossbar. Namely, NoC fabrics present good scalability features, port speed and path diversity [151]. The literature is full of proposals that have adopted the NoC concept for high-performance packet-switching. Descriptions for NoC-based Ethernet switches are provided in [113, 152]. The Unidirectional

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

NoC crossbar switch fabric (UDN) was presented in [151] [153]. The MDN switch [154] was later introduced as an extension of the UDN proposal. More recent results [155] suggested an implementation of a single-stage crossbar fabric with NoC-enhanced FPGA and different routing algorithms. The related work is abundant. However, the application of NoC-based crossbar fabrics has been restricted to single-stage switches, despite their high practicality and potential. This motivates the following idea:

✱ **Idea.** *“Changing the buffered middle-stage modules from passive blocs of memories into small active networks, would benefit the whole multistage switch in terms of HW and scheduling requirements”.*

In the following sections, the switch architecture, packet buffers, and the scheduling algorithms are overviewed.

4.2 Clos-UDN switch architecture

In this section, a three-stage Clos-UDN switch architecture is described with an emphasis on the NoC based central modules. Then, the dispatching process meant to transfer packets to the middle stage is introduced.

4.2.1 The switch model

The reference design of a Clos-UDN switch of size $N \times N$ is depicted in Figure 4.1. The key notations used in this chapter are listed as in TABLE 4.1. The first stage of the Clos-UDN comprises k IMs, each of which is of size $n \times m$. The second stage is made of m UDN fabric modules, each of dimension¹ $k \times k$. The third stage consists of k OMs, each of which has $m \times n$ dimension. Although it can be general², the proposed Clos-UDN architecture has an expansion factor

¹Unlike conventional Clos-networks, the central modules of the Clos-UDN can be of size $k \times M$ crosspoints, where M refers to the NoC depth and $M \leq k$.

²The Clos-UDN can of course be of any size, where $m \geq n$. This would simply require packets insertion policy in the FIFOs should one need to maintain low-bandwidth FIFOs. This is considered to be out of the scope of the current work.

4.2 Clos-UDN switch architecture

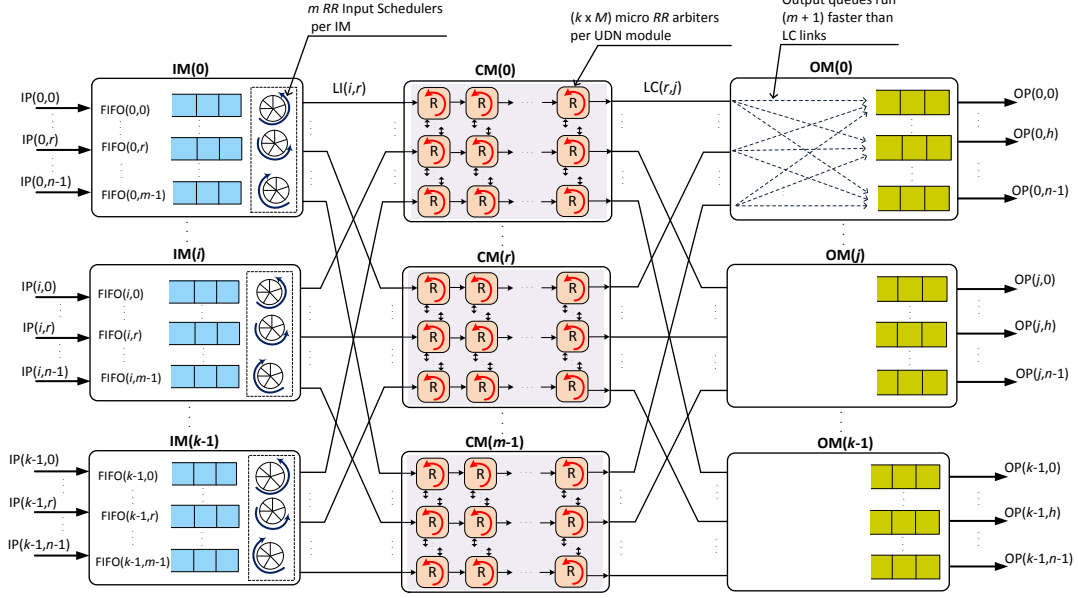


Figure 4.1: $N \times N$ three-stage Clos-UDN packet-switch architecture with dynamic dispatching scheme.

$\frac{m}{n} = 1$, making it a *Benes* lowest-cost practical non-blocking fabric. An $IM(i)$ has m FIFOs each of which is associated to one of the m output links denoted as link $LI(i, r)$. An $LI(i, r)$ is related to a $CM(r)$. Because $m = n$, each $FIFO(i, r)$ of an input module, $IM(i)$, is associated to one input port, $IP(i, h)$, and can receive at most one packet and send at most one packet to one central module at every time-slot. A $CM(r)$ has k output links, each of which is denoted as $LC(r, j)$ and is connected to $OM(j)$. The output module $OM(j)$ has n OPs, each of which is denoted $OP(j, h)$, and has an associated output buffer. The output buffer can receive at most m packets and forward one packet to the output line at every time-slot. Packets destined to different output ports are accepted to the NoC fabric even when some outputs are busy with other packets.

4.2.2 NoC based central modules

The key element of the current design, is the UDN [151] modules that are plugged into the Clos-network. Figure 4.2 shows a simple diagram of the UDN

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

Table 4.1: *The terminologies for the Clos-UDN switch.*

Notation	Description
$IM(i)$	$(i + 1)^{th}$ IM at the first stage
$CM(r)$	$(r + 1)^{th}$ CM at the second stage
$OM(j)$	$(j + 1)^{th}$ OM at the third stage
i	IM number, where $0 \leq i \leq k - 1$
r	CM number, where $0 \leq r \leq m - 1$
j	OM number, where $0 \leq j \leq k - 1$
h	IP/OP number in each IM/OM, respectively, where $0 \leq h \leq n - 1$
$IP(i, h)$	$(h + 1)^{th}$ IP at $IM(i)$
$OP(j, h)$	$(h + 1)^{th}$ OP at $OM(j)$
$FIFO(i, r)$	First-In-First-Out queue that stores packets going to CM module, r .
$LI(i, r)$	Output link at $IM(i)$ that is connected to $CM(r)$
$LC(r, j)$	Output link at $CM(r)$ that is connected to $OM(j)$

switch. Every central unit in the Clos-UDN switch is a two-dimensional mesh $k \times k$ of small on-chip packet-switched input-queued routers that transport packets across the NoC in a multi-hop fashion. All on-chip routers have small input FIFO queues. These queues have a predefined capacity, that is called *Buffer Depth* – BD . They are employed to store packets on their journey to their outputs. To avoid elastic buffers, a credit-based flow-control is implemented, and

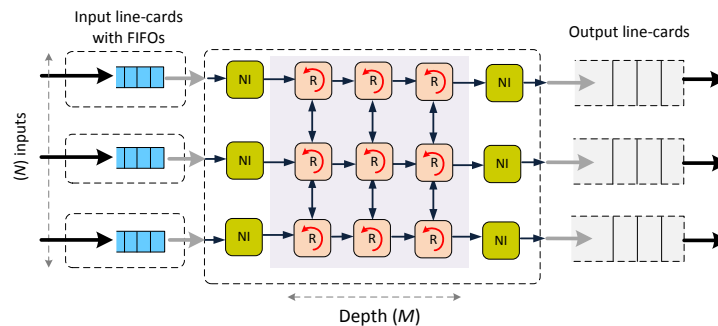


Figure 4.2: *The UDN crossbar switch.*

packets are only sent to a downstream router when a buffer space is available [123]. Packets have a fixed-size, with their relative routing information being

4.2 Clos-UDN switch architecture

stored at the header part. When they enter the NoC layout, packets are fully received and stored in one of the router's buffers before going to the next hop. Using a deadlock-free routing algorithm, named “*Modulo XY*” [154], packets progress throughout the NoC fabric at a rate of one packet per time-slot [154]. In the following parts, the term speedup (SP) will be used to refer to the speed ratio at which the on-chip links of the NoC modules can run with respect to the LI/LC links speed. It is equivalent to the NoC fabric removing up to SP packets from one input, and sending up to SP packets to one output per time-slot. On the scheduling front, the on-chip routers make local decisions about the packets' next destinations using a RR arbitration, making the scheduling process distributed.

As subsequent parts shall demonstrate, the Clos-UDN switch can sustain high throughput and low delays under heavy loads if the fabric is running with a small speedup ($SP > 1$). Given the small sized on-chip routers and the short wires, a speedup of two can be readily affordable. Section 4.3 and Section 4.5 further study this property.



“Modulo XY” on-chip routing algorithm

NoCs are the network version of SoCs where packet-switching and routing algorithms are key terminology. On-chip routing defines the way packets peruse their path from a source node to a destination node. The popular “*XY*” routing have been long used in NoC systems for its simplicity. It was introduced by Wang Zhang and Ligang Hou [156] for mesh and torus networks. Every router in position (x, y) compares the packet's destination (x', y') with the current coordinates, and adjudicates the routing direction. In the general case, if $(x' > x)$, the packet moves *East*, else it goes *West*. When reaching the line where the corresponding destination node is found, a packet moves *North* if $(y' > y)$, and *South* if $(y' < y)$. The “*XY*” in its basic form poorly distributes traffic across the network as it tucks the biggest load in the middle of the NoC [157]. Many algorithms were proposed to better equalize the load over the network. The “*Modulo XY*” introduces an extra turn in a column before the one where is located the destination node. To each flow* is associated a fixed turn index t that is calculated as in Algorithm 1.

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

Algorithm 1: “Modulo XY” routing

Require:

(x, y) : The current node coordinates
 (x', y') : The destination node coordinates
 N : Number of I/Os of the NoC
 M : Depth of the mesh

```
1: Switch (packet input buffer) do
2:   Case North
3:     if ( $x = x'$ ) then East, else South
4:   Case South
5:     if ( $x = x'$ ) then East, else North
6:   Case West
7:     if  $((x' \bmod M) = (N - x + y + t) \bmod M)$  then
8:
9:       if ( $y' > y$ ) then South, else North
10:    else
11:      East
```

*A flow is defined to be the set of packets that arrive in consecutive time slots, and which have the same input and output tuple.

4.2.3 Head-of-Line blocking

Common single-hop crossbar switches experience HoL blocking whenever packets wait in input line cards for their corresponding output ports to become available. In an IQ switch, packets at the head of the input FIFOs block other cells that are queued behind them, even though the latter are destined to free output ports [151]. Multi-hop NoC-based crossbars such as the UDN modules, does not suffer the HoL limitation because of the multistage and pipelined nature of the NoC itself. Packets from a single input port heading to different output ports can be accepted into the NoC structure even if their outputs are busy serving other cells. Figure 4.3 gives a time-space diagram to show how packets progress throughout the NoC fabric. In addition to the geometric features of the NoC-based switching fabrics (path diversity), implementing adequate routing methods is liable to enhance the load-balancing. Consequently, the traffic might be parallelised on multiple paths, and packets which intend to go to different output ports interfere little with each other. In summary, simple FIFO queues

can be used instead of the costly VOQs on the input line cards with no severe performance degradation [158].

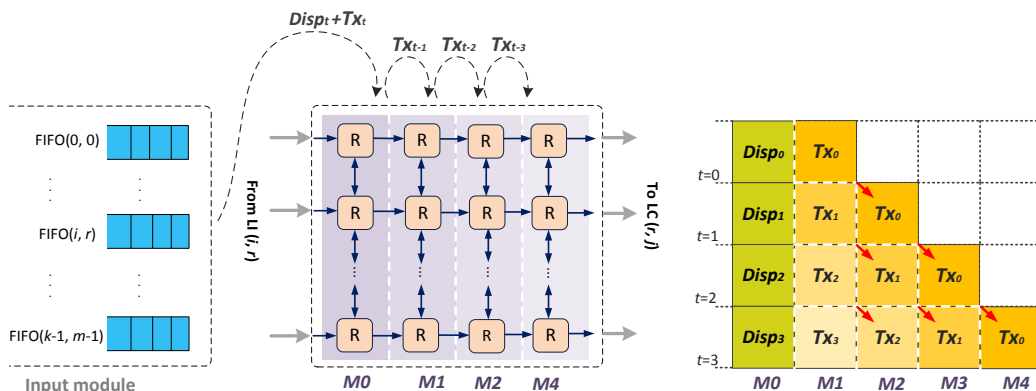


Figure 4.3: Pipelined working of Clos-UDN dispatching and packets forwarding through UDN modules.

4.2.4 Packet dispatching and scheduling in multistage switches

Packet scheduling is as important as the architectural design of the switch. It strongly depends on the characteristics of the fabric and the packet buffers disposition. In bufferless multistage switches such as the MSM, the ultimate purpose of the packet scheduling process is to establish a conflict-free matching. This mandates two types of matching [64, 69, 70]. The first one takes place within an input module. It ends up selecting the eligible queues among non-empty candidate VOQs. The second matching happens between IMs and CMs. A previous work has shown that both of these matchings are quite complex and time consuming for two reasons: The high number of input queues per IM for the first matching, and the global synchronization needed to match the IM-CM ports. All of these, impose a big dull on the scheduling system.

Introducing buffers to all stages of a Clos-network switch helps absorb the contention and obviates the need for a complex and centralized matching over multiple links and modules. It was found that larger internal buffers enhance the performance of the MMM switch, especially under heavy unbalanced input loads.

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

However, the modern integrated circuit technology still limit concrete buffered fabrics production. At the scheduling front, the MMM switch as described in [74] employs a single central admission scheduler. This scheduler manages all credit and grant sub-schedulers that operate in a pipelined way. The arbitration process works as follows: Schedulers at the input line cards send requests to the central arbiter. Credit schedulers associated to the switch outputs manage as many credit counters as crosspoints and allocate buffer space for packets. The grant schedulers select one among the candidate requests, and send back a grant signal to the central admission scheduler.

The proposed Clos-UDN switch greatly simplifies the process of packet dispatching and scheduling. First, each IM needs to maintain only m input queues and each input port of an input module can send packets to only one FIFO queue per time-slot. Consequently, running the input FIFOs at only twice the line rate is sufficient. In the Clos-UDN switch, there are m input schedulers in every IM, one per FIFO queue. The RR input schedulers are initialized to different values and they keep updating their selection pointers to one position at the end of every time-slot. This guarantees that all pointers are always desynchronized and that no conflict in the LI links selection is likely to happen. At the start of every time-slot, a scheduler selects an LI link among m links in a RR fashion. Then, the HoL packet is transferred from the input FIFO to the corresponding CM. A packet is accepted to the CM module if the left-most NoC router still has room in its left buffer. Once at the NoC, the “*Modulo XY*” routing algorithm takes over, and routes the packet to its outgoing LC link.

The Clos-UDN switch differs from the bufferless and buffered variety of multistage switches. This is mainly attribute of the switching fabric architecture that do not require any inter-module matching. The central stage NoC fabric allows a parallel and distributed packet forwarding. Mini-routers of each UDN fabric decide independently about the next hop of the packets. They continuously examine and modify the route information until the packet reaches its LC destination. Whenever packets contend for a link, those who loose the arbitration process would remain stored at the router’s buffers before they are granted access to the shared resource (on-chip link) [151, 153]. Correspondingly, the $LC(r,j)$ links contention gets resolved within the UDN units as packets

progress throughout the NoC, as Figure 4.3 shows. Overall, the process of path-allocation in the Clos-network is relaxed. More importantly, no centralized global decision and no synchronization are needed.

4.3 Hardware requirements

In the previous section, the packet scheduling process in the MSM, MMM, and Clos-UDN multistage switching architectures were described. The CRRD scheduling scheme and its derivatives, as adopted for MSM, perform iterative matchings between the set of requesting VOQs and available LI links. Note that in the MMM switch, no such process is needed. Packets are simply selected¹ to move from the input queues at the IMs to the available crosspoint buffers. Upon their arrival to the middle-stage switching fabric, packets undergo what is called the *output scheduling*. This process approves a set of packets to be transferred to their corresponding output buffers at the OMs. In this section, the hardware requirements of the proposed Clos-UDN switch are briefly compared to those of the MSM switch and the MMM switch. TABLE 4.2 compares some features of the three aforementioned switching architectures.

A prototype of the UDN crossbar switch has been synthesized in a previous work [153] using an ASIC 65 nm CMOS technology. The synthesis of a 3×3 UDN switch with no optimization measures, achieved 413 MHz with a cell area of 4.8 mm^2 including the Network Interfaces (NIs). As defined in the same work, the degree of a router is the number of its I/O ports. The UDN fabric has a degree 3 (in the *East* and *West* mesh columns) and a degree 4 (in the intermediate mesh columns) NoC routers that occupy respectively 0.29 mm^2 and 0.38 mm^2 . The NIs occupy 0.32 mm^2 considering the same synthesis technology [151]. Namely, registers that are used for the routers' FIFO queues dominate the area. The die area of the circuit is shown to drastically shrink if dedicated hardware rippled-through FIFOs and other CMOS process technologies are used

¹The first step of packet scheduling in a fully buffered three-stage Clos-network switch is generally quoted input scheduling. During this phase a particular selection logic might be adopted to choose the way to serve packets from the input queues at the first stage of the switching network, to the middle stage fabric.

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

Table 4.2: Compare the HW requirements of MSM, MMM and Clos-UDN switches.

	MSM [64]	Three-stage Clos-UDN	MMM [75]
Input buffers per IM	$(n \cdot k)$ VOQs	m FIFOs	$(n \cdot k)$ VOQs + $(n \cdot m)$ VCMQs
Central modules	Bufferless	NoC-based UDN	buffered
Dispatching scheme	CRRD	RR dispatching	RR / LQF
Contention resolution	Two phase matching	NA	NA
IM arbiters	$N + m$	m	$n + m$
CM arbiters	m	NA	k
Scheduling algorithm	Centralized/complex	Parallel / distributed	Distributed
In-order packets delivery	Yes	No	No
Parametrization	NA	YES	YES
Performance / scalability	low	High	Good

(e.g. The area of an $N = M = 32$ switch using CMOS 65 nm process is about 403 mm^2 and only 134 mm^2 if 90 nm CMOS technology is used). Adopting a 65 nm CMOS process, a central module of size 8×8 used in a 64×64 Clos-UDN network switch would occupy: $0.29 \times 2M + 0.38(N - 2)M + 0.32 \times 2N \approx 224 \text{ mm}^2$ per CM. In the following part, a rough estimate of the the dispatching time complexity in the MSM, MMM, and Clos-UDN switches is given. Also, an estimation for the hardware complexity of packet dispatching and scheduling in the three architectures is provided.

4.3.1 Dispatching time

To provide better insight into the complexity of every design, a rough estimation of the time and hardware complexity of every switch is provided. The MSM switch runs with the CRRD algorithm which is a two-phase matching process: Matching within IMs and the IM-CM matching. The first phase is an iterative matching that runs up to $iter$ times to maximize the number of connected VOQs to the LI links. At every iteration, two RR arbiters are used as follows: Every link LI selects one out of at most $(n \cdot k)$ requesting VOQs. A VOQ arbiter chooses one among at most m grants. Hence, the resulting complexity in this phase is $\mathcal{O}(iter (\log(nk)))$ where $iter$ is the number of iterations ($1 \leq iter \leq m$). The

4.3 Hardware requirements

second phase involves the matching between the set of IMs and CMs, during which every $LC(r, j)$ arbiter chooses one among at most k requests. The time complexity of this phase is then $\mathcal{O}(\log k)$.

The asset of a fully buffered Clos-network switch is the *relatively* simple¹ packet scheduling mechanism. Buffers are more like terminals where transiting packets spend some time waiting for a resource to become accessible. The more terminals are available, the less complex is the task of packets forwarding. Arriving packets get stored in VOQs at the input line cards of the MMM² switch. There is a total of N arbiters, one per input port, to select the cell to send to one of the m Virtual CM queues, VCMQs, at the IMs [75]. The selection of VCMQs is RR based. A total of m arbiters in each IM are used to perform the selection of the CM through which the cell is sent. Then, the dispatching complexity³ is $\mathcal{O}(\log (nm))$.

Although there are some similar points between the MMM switch and the Clos-UDN switch, packets are managed in completely different ways. In the Clos-UDN switch, the packet dispatching scheme is non-iterative. At each cell-time, m RR arbiters in the IMs select CMs to dispatch the HoL packets. This makes the complexity time equal to $\mathcal{O}(\log m)$. The dispatching process and packets routing through the UDN modules work in parallel. The pipelined nature of the UDNs makes the dispatching time at time-slot t ($Disp_t$) and the packet scheduling and forwarding through the NoC (Tx_t) overlapping. The flow of packet dispatched to a particular UDN module at time-slot $t = 0$ is called F_0 . As depicted in Figure 4.3, F_0 arrives to the NoC routers of the first column M_0 . Forwarding decisions are taken, and packets are transferred to inputs of the next hop. At time-slot $t = 1$, a new flow of packets F_1 arrives to M_0 , while F_0 gets routed to the next stage of the UDN. Note that maintaining static connections between the two first stages of the Clos-UDN switch removes the $k \times m$ input schedulers used to dynamically dispatch cells to the central modules. Henceforth, the complexity of the switch is considerably reduced.

¹Compared to matching algorithms in bufferless multistage switching architectures.

²MMM packet-switch architecture as described in [75].

³As only non-blocking Clos-networks are considered, $m \geq k$.

4.3.2 Hardware complexity for packet dispatching

The hardware dispatcher is responsible for managing the release of the HoL packets from currently matched input queues. Keeping the packet dispatching plain has other benefits besides the hardware requirements. Actually, a dispatch delay that is within the resolution of the application simplifies the implementation, and averts the need for clock synchronization [159]. In what follows, the hardware requirements for packet dispatching for the MSM, MMM and Clos-UDN switches – respectively – are specified.

For the MSM switch, every IM has m output-link arbiters and N VOQ arbiters. Generally, the complexity of a RR arbiter is $\mathcal{O}(\log n_{req})$, where n_{req} is the number of requests to be selected by the arbiter. The hardware complexity of an IM module in an MSM switch with a CRRD scheme is $\mathcal{O}(\log (mN))$ [64]. There are also k $LC(r, j)$ arbiters at the central modules, each of which selects one request out of at most k requests. The total complexity of a CM is $\mathcal{O}(\log k)$. During the first phase of the CRRD matching, requests are sent to the LI arbiters which in return send back grants to the selected VOQs. A VOQ arbiter accepts one among the received grants. Obviously, the size of the interconnect between the m output links and the $(n \cdot k)$ VOQs arbiters increases with the switch size. This makes the wiring more complex. Moreover, the number of crosspoints N_{xp} for the interconnection wires set between the IM arbiters is given by $N_{xp} = \frac{3}{4}nkm(nk - 1)(m - 1)$ [64]. Given the excessive demands of the bufferless architecture, the CRRD dispatching scheduler was suggested to be implemented on multiple-chips [64]. This would diminish the layout complexity. Still, the interconnection between the chips on the *Printed Circuit Boards* as well as the number of pins in the scheduler chips would become higher and more expensive.

Unlike the MSM switch, the MMM architecture requires many buffers to house packets at every stage of the Clos switch. In addition to the N VOQs present at the input ports, each IM block is made of $(n \cdot m)$ VCMQs. The operation of VOQ and VCMQ selection performed before the phase of packet dispatching, makes the hardware complexity of an IM – roughly – equal to

4.4 Resolve the out-of-order packets delivery

$\mathcal{O}(\log(nN))$. Using the same previous estimation logic, the HW complexity of a CM and OM are $\mathcal{O}(\log k^2)$ and $\mathcal{O}(\log(nm))$ – respectively.

As for the Clos-UDN switch, there are m arbiters per IM which are associated to the m FIFOs. A queue arbiter selects one among m CMs to dispatch the current HoL packet which makes the hardware complexity of an IM equal to $\mathcal{O}(\log m)$. Every CM bloc at the central stage is made of $(k \cdot M)$ mini-routers. Every on-chip router selects packets in a RR manner to forward them to the next hop. This results into a complexity of $\mathcal{O}(\log(kM))$. The Clos-UDN switch has a simple dispatching scheme and the IM arbiters are not connected to any others. This saves the need for complex interconnects. Such a feature partially decouples the scheduling function from the HW, and makes the Clos-UDN scalable independently of the switch size. The contention resolution in the Clos-UDN switch is progressively resolved as packets advance through the central modules, and the implementation complexity is considerably reduced. In [153], a HW implementation of a single-stage UDN packet-switch is proposed where it is shown that a UDN module is perfectly feasible considering the current technology, and that a cost/performance trade-off can be made by varying the switch parameters and/or the synthesis technology.

The Clos-UDN switch falls within the category of buffered multistage switches, where in-order packet delivery is a critical issue. In this chapter, the resequencing mechanisms are merely amended by configuring the inter-stage connections of the Clos-network.

4.4 Resolve the out-of-order packets delivery

Pure space switching fabrics like S^3 introduce too much competition between cells at all stages. They suffer low throughput and have crippled performance. Introducing memories to packet-switching architectures has always been the remedy to resolve contention. So far, buffreless architectures need complex and expensive matching mechanisms to make the path decision. However, they guarantee an ordered packets delivery. In the opposite case, the fully buffered architecture MMM has good performance and simpler scheduling process. Yet, packets experience variable queuing delays in the internal buffers leading in a

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

mis-sequenced packets transmission [160]. Ordering packets can be costly in terms of memory and processing time [74, 77, 86]. Unfortunately, the proposed Clos-UDN with RR packet dispatching scheme suffers the same limitation. Packets are dynamically sent to the UDN central modules where they get serviced with variable delays.

✱ **Idea.** *“Removing the $(n \cdot m)$ input schedulers, and putting in place a static configuration of the connections between the IMs and CMs saves the in-order packet delivery. Since, a deterministic on-chip routing algorithm is used, packets of the same flow are consistently forwarded through the same path.”.*

A static configuration of the IM-CM connections is proposed. The new switching architecture can be viewed as a two-stage network. The intuition behind the two-stage approach is as follows. It is known that the “*Modulo XY*” algorithm used to route packets inside the UDN modules is a deterministic algorithm that ensures that packets of the same flow do follow the same path throughout the NoC fabric. Unlike previous proposals, using a static IM-CM connections configuration entails no re-sequencing buffers, no synchronization signals and no complex re-ordering methods. Theoretically, a two-stage Clos-network is only rearrangeably non-blocking. As path relocation is prohibited in practical Clos switches, the two-stage architecture becomes blocking and compromises the switch throughput [161]. All the same, two-stage interconnects prove to be interesting mainly for optical switches [77, 162, 163]. The architecture improvements from the single-stage baseline are still significant. Although, the two-stage Clos-UDN diminishes the extra switching area per IM/OM pair, it contributes toward reducing the cost/complexity. More importantly, it ensures an ordered packets delivery (see TABLE 4.3). The switch is still scalable thanks to the scalable central modules and presents good features to build high-radix switches. The next section is reserved for the performance evaluation of the proposed Clos-UDN architecture and to examine the response of the switch under various traffic types.

Table 4.3: *HW requirements: Dynamic vs static dispatching in Clos-UDN packet-switch.*

	Three-stage Clos-UDN	Two-stage Clos-UDN
Dispatching scheme	Dynamic RR dispatching	Static dispatching
IM arbiters	m	NA
In-order packets delivery	No	Yes
Performance / scalability	High	Good

4.5 Performance evaluation

An event-driven simulator is used to evaluate the performance of the three-stage Clos-UDN with a dynamic and a static dispatching schemes. Unless it is mentioned, the switch size is set to 64×64 , the on-chip buffers – BD – are worth of four packets each, and square UDN meshes are considered for the central stage of the Clos-network (full mesh depth for which $M = k$). In the simulation figures, $iterx$ stands for x number of iterations for the CRRD dispatching algorithm and SPx means that the on-chip links of the UDN modules in the Clos-UDN switch run x times faster than the LI/LC links. As for MMM, $xbuff$ is the size of the middle stage’s crosspoint buffers. The performance of the Clos-UDN switch is compared to the MSM switch architecture with CRRD dispatching as described in [64] (used as a baseline) and the MMM switch as it was described in [77]. Both architectures are compared for various switch sizes and traffic scenarios. It is important to note that the speedup used in the Clos-UDN is different than the conventional speedup [17], where SP refers to the internal switch over-speed factor with respect to the external line rate. Here, SP refers to the over-speed factor of *only* the NoC routers inside each UDN central module, excluding the LIs and LCs. Meaning, just like the MSM switch, the Clos-UDN always sends at maximum one packet per LI/LC link per time-slot. Since the Clos-UDN does not use iterations (i.e. time) in its matching, this could be compensated by internally running the UDN CMs with small SP values.

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

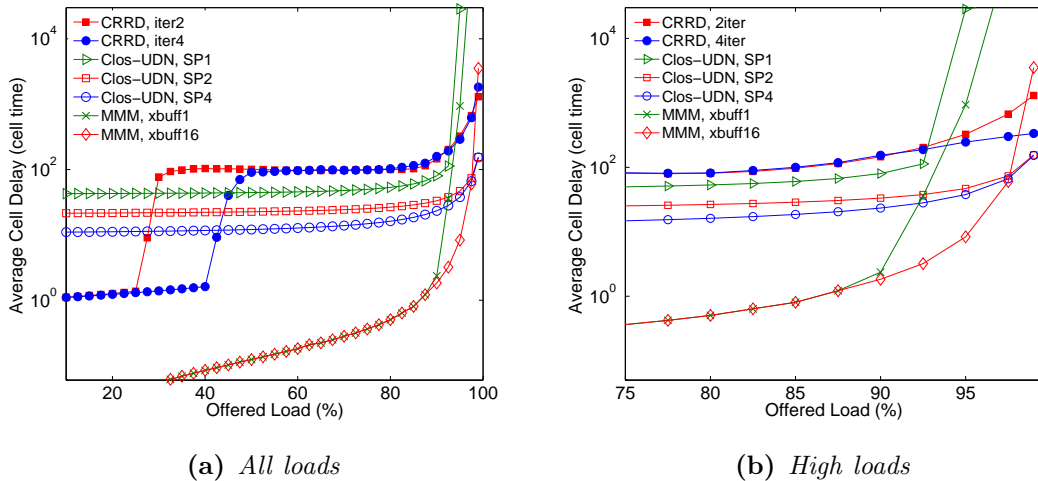


Figure 4.4: Delay performance for a 256×256 switch under Bernoulli uniform traffic.

4.5.1 Clos-UDN versus MSM and MMM

In this part, the throughput and latency performance are measured under various synthetic traffic patterns. The robustness of the design to traffic and switch size variation is also examined.

Bernoulli uniform traffic

Figure 4.4(a) compares the packet delay performance of the Clos-UDN using different speedup values, to that of MMM switch and MSM switch employing CRRD with different iterations. The Clos-UDN switch is tested with 4-packets large buffers at the on-chip routers' inputs making a total of 12 packets buffering per crosspoint. Running the CM units at a speedup factor of one makes the switch achieve up to 90% throughput. It is the packets progressing in the central switching units by one at each cycle ($SP = 1$) that prevents the switch from achieving full throughput. Note that increasing the speedup factor to two suffices for the switch to achieve full-throughput. The Clos-UDN architecture outperforms the MSM under medium-to-high uniform traffic arrivals (which are more relevant in the context of DCNs). The slightly higher delay experienced by the Clos-UDN switch under light loads is due to the time required to fill-in

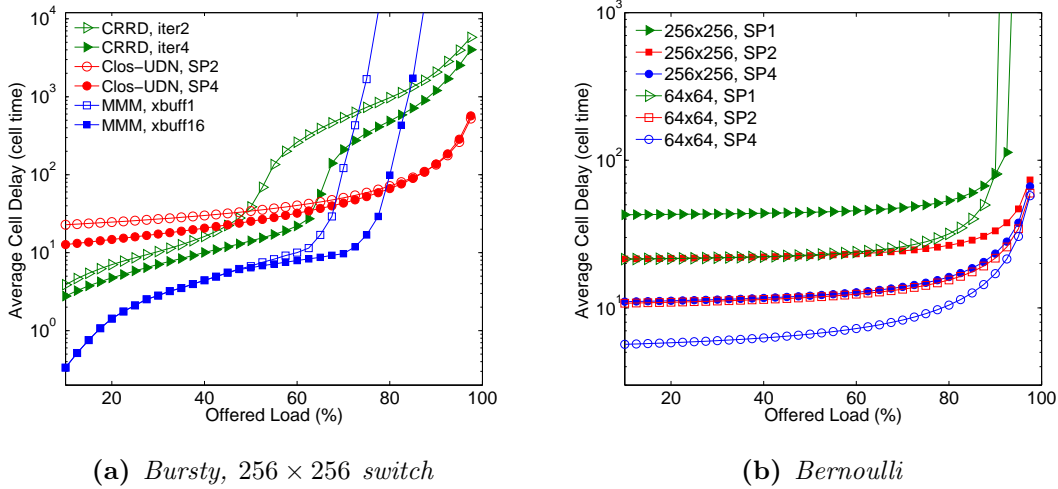


Figure 4.5: Performance of Clos-UDN and MSM switches, under uniform traffic.

the pipeline of the multi-hop NoC fabric, as shown in Figure 4.3. However, the Clos-UDN maintains moderate and almost constant delay irrespective of the traffic load. When the load is larger than 0.9, the delay performance of the Clos-UDN with $SP = 2$ becomes better than the MSM switch running the CRRD algorithm 4 times (iterations) and the MMM as Figure 4.4(b) shows. By increasing the speedup of the UDN switch, the average cell delay is pulled down. The MMM switch outperforms the MSM and Clos-UDN switches if light to medium loads strike the switch inputs. Setting the crosspoint buffers' size to $b = 1$, makes the MMM throughput – almost – equal to 94%, and full throughput is achieved if the buffers are as large as $b = 16$. Clearly, the Clos-UDN deals in a better way with high loads for which it keeps the lowest system delay (compared to the MSM and MMM architectures, under high loads).

Uniform bursty traffic

High-bandwidth demanding applications make the bursty traffic pattern prevalent in a data center network with high-levels of peak utilization. In what follows, the effect of traffic burstiness on the Clos-UDN switch is inspected. The default burst length is set to 10 packets. Figure 4.5(a) reveals that the delay's growth in the Clos-UDN switch under bursty arrivals, is smoother than that of the

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

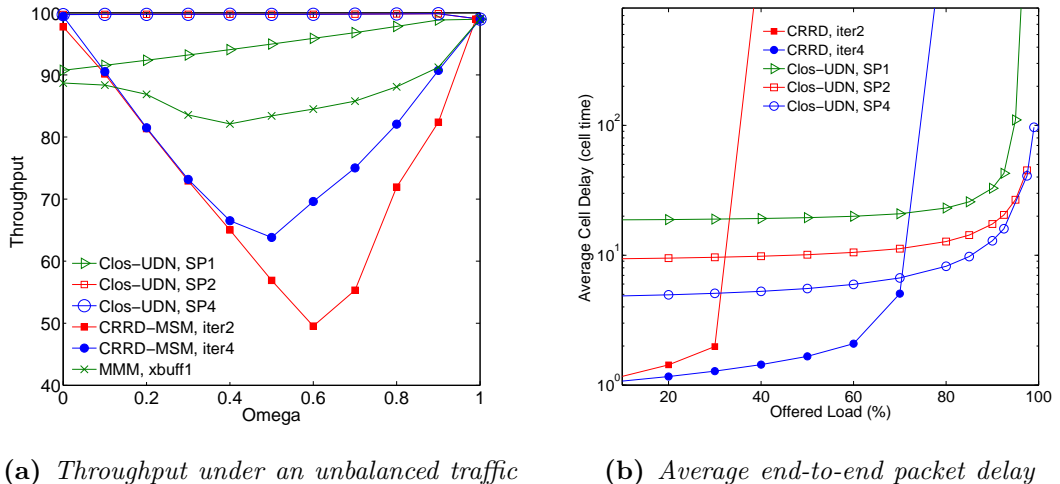


Figure 4.6: Performance of a 64-ports switch under unbalanced traffic.

MSM switch. Increasing the number of iterations for the CRRD provides better matching between IMs and CMs, and resolves faster the contention. This leads to improved switch performance when the load is below 0.7. However, the MSM switch is instable and performs poorly even under uniform traffic when the load becomes important. Increasing SP reduces the initial delays for the Clos-UDN. Simulations show that the MMM is less efficient when evaluated for bursty traffic. Although the switch has lower average packet delay, the Clos-UDN switch with a minimum SP = 2 proves to outperform both MSM and MMM under heavy bursty traffic arrivals. Visibly, MMM has degraded throughput and increasing the crosspoint buffers to $b = 16$ is of little effect, as it only shifts the switch throughput from 77% to up to 86%.

Unbalanced traffic

Next, the Clos-UDN switch is assessed under non-uniform traffic [64]. Previous results are reproduced for the MMM switch [77] implementing the Longest Queue First (LQF) selection at the input ports and a RR arbitration in the different modules. Figure 4.6(a) depicts the switch throughput when the unbalancing coefficient ω is varied. The Clos-UDN with SP = 1 achieves up to 90%

throughput for $\omega = 0$ (uniform traffic), as has been already shown in Figure 4.4(a). The MMM switch achieves better throughput than the MSM switch (60% throughput under hot-spot traffic if 4 iterations are used). However the Clos-UDN switch has higher and more stable throughput variation than both the semi-buffered and the fully buffered architectures across the whole range of ω . Setting $\omega = 0.5$ corresponds to a hot-spot traffic, where 50% of the input load goes to one output while the rest is equally distributed over the remaining outputs. A further step in analysing the Clos-UDN switch performance consists on inspecting the average delay under non-uniform traffic pattern in comparison to MSM. Figure 4.6(b) depicts simulation results for a 64×64 switch size. Curves in Figure 4.6(b) point out that the Clos-UDN switch architecture has much better average delay than the MSM, irrespective of the Clos-UDN speedup and the CRRD number of iterations.

4.5.2 Further analysis of the Clos-UDN switch

In this subsection a set of parameters of the Clos-UDN architecture are varied to study the effect of each one on the overall switch' performance.

Varying the switch size

The performance curves depicted in Figure 4.5(b) show that increasing the Clos-UDN valency has a minor effect of the overall delay, making it truly a scalable solution and a good alternative for DCN switches. Large switches can achieve good performance if the speedup factor is increased to just 2. Interestingly, a 256×256 Clos-UDN switch running at a speed $SP = 4$ has an average cell latency that is approximately the same as 64×64 switch with $SP = 2$.

Varying the depth of the UDN modules

The Clos-UDN is configurable. Changing the number of the UDN's intermediate stages – M – can be done to trade-off the design cost to the overall system performance [151, 153]. However, this cannot be done without limits, as it may cause the structure of the NoC to be congested and the performance to collapse. The Clos-UDN's initial latency is acquired from the multi-hop nature of the NoC

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

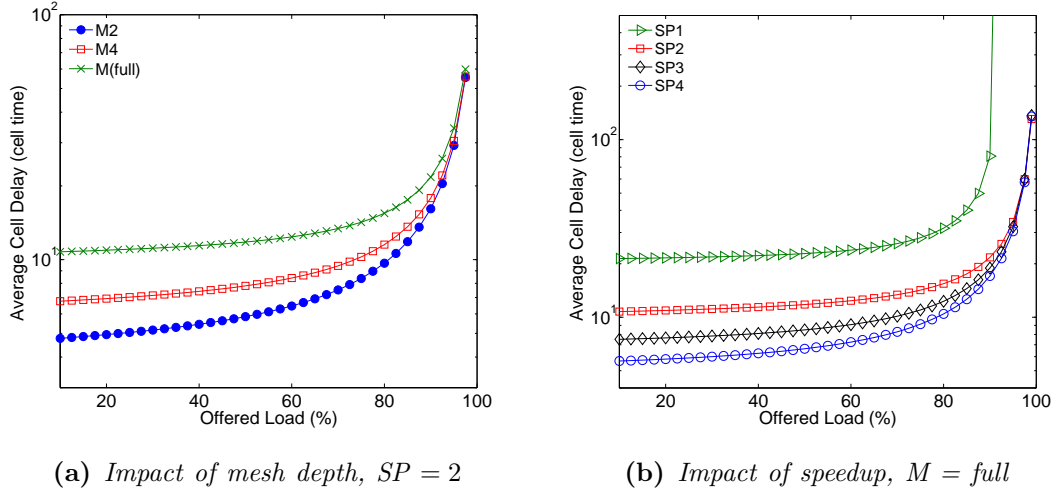


Figure 4.7: Switch performance, under Bernoulli traffic.

fabric. In conventional crossbars, packets cross the fabric in one shot. However, in a UDN module, they have to cross at least M on-chip routers to reach their destination, which results in a cumulative delay. Reducing M causes the packets to travel through less intermediate stages before arriving to LC links. Hence, the average packets latency gets lower when the switch is non-congested when the NoC fabric is running at a minimum $SP = 2$, as Figure 4.7(a) shows.

Running the central modules faster

As Part of the Clos-UDN architecture, all embedded routers of the CM blocks are input-buffered routers. They require speedup to achieve full-throughput. Figure 4.7(b) shows that increasing SP contributes towards diminishing the overall switch latency when the switch is not congested. Reducing the number of columns – M – to only two improves the system’s latency under light traffic loads. Figure 4.8 shows that decreasing M is more effective than running the CMs faster (increasing SP for a given depth M). In conclusion, running the UDN central modules faster for a given mesh depth has less impact than dropping $(k - M)$ columns. All the same, one can choose the best settings, for the Clos-UDN architecture to achieve pre-estimated performance levels.

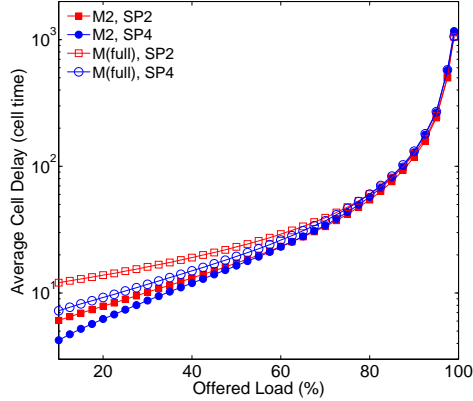


Figure 4.8: Impact of the speedup SP , and mesh depth M , on the Clos-UDN switch latency, Bursty traffic.

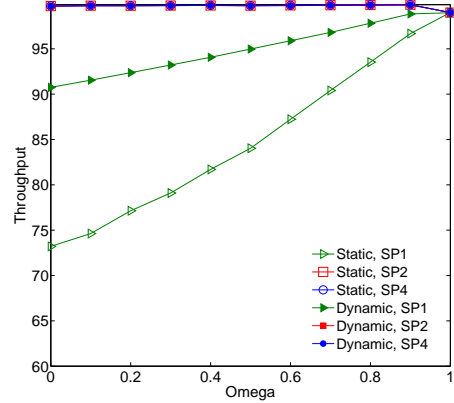


Figure 4.9: Throughput stability of the two-stage and three-stages Clos-UDN under Unbalanced traffic.

Changing the buffer depth

A single mini-router in a UDN module is a complete switching element with small input memories and a processing unit. The input buffers account for the major part of a router’s area. Therefore, they need to be as small as possible to reduce the design cost. Increasing the BD improves the system latency, but reducing the buffering amount can produce problems related to insufficient buffering [70].

4.5.3 Performance of the two-stage Clos-UDN switch

In the following set of simulations, a 64-ports switch with a full mesh depth ($M = k = 8$) and a variable speedup factor is considered. The terminology dynamic and static is used to present the Clos-UDN performance with a dynamic RR dispatching scheme and a static packet dispatching process – respectively. There are many ways to configure the first two stages connections. The default Clos interconnection is considered as Figure 4.10 depicts. Considering a directional traffic, the “Modulo XY ” algorithm would work as follows: All packets that come from an input $FIFO(i, r)$ heading to $OP(i, h)$ are always forwarded to the ingress i of $CM(r)$ via $LI(i, r)$. Resolving the packets destination would result in

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

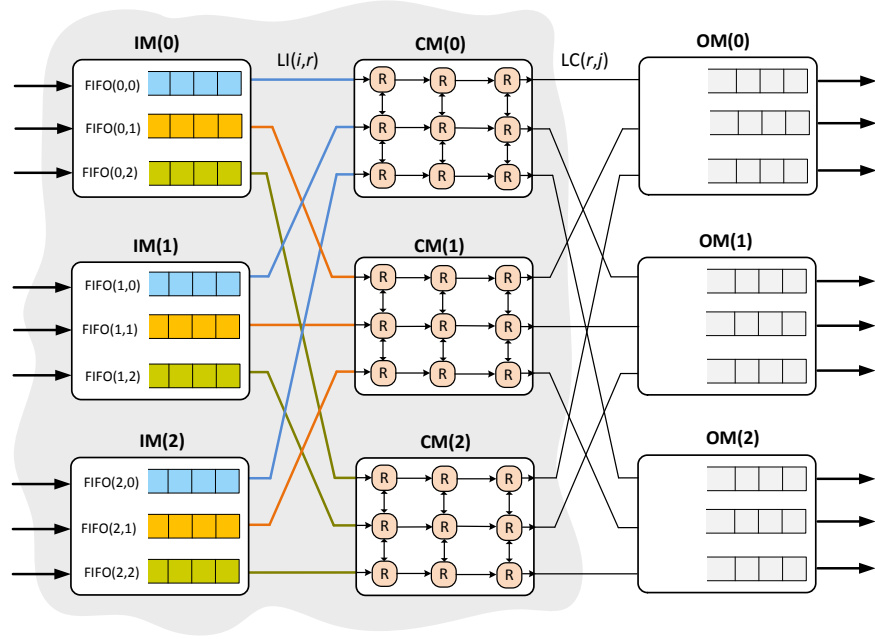


Figure 4.10: An example of a 9×9 Clos-UDN switch with static configuration of the IMs/CMs interconnections.

a direct route across the UDN fabric (the route from an input to an output of the NoC mesh with no turns). In case of crossing traffic, where packets stored in $\text{FIFO}(i,r)$ are destined to $\text{OP}(j,h)$ ($j \neq i$), the load is distributed in the UDN fabric and forwarded using different paths to the right LC link.

Uniform traffic

Under uniform packet arrivals, the two-stage Clos-UDN switch offers good throughput and tolerable average packet delays. Figure 4.11(a) compares the delay performance of the three-stage Clos-UDN and the two-stage Clos-UDN with in-order packets delivery guarantee. By adopting a static dispatching scheme, the number of links that are simultaneously available between any pair of IM/OM is reduced. This accounts for a throughput degradation that one can notice when the central switching modules are run at $\text{SP} = 1$. Speeding up the CMs preserves the switch throughput, and a $\text{SP} \geq 2$ makes the two-stage Clos-UDN achieve performance comparable to that of a three-stage Clos-UDN

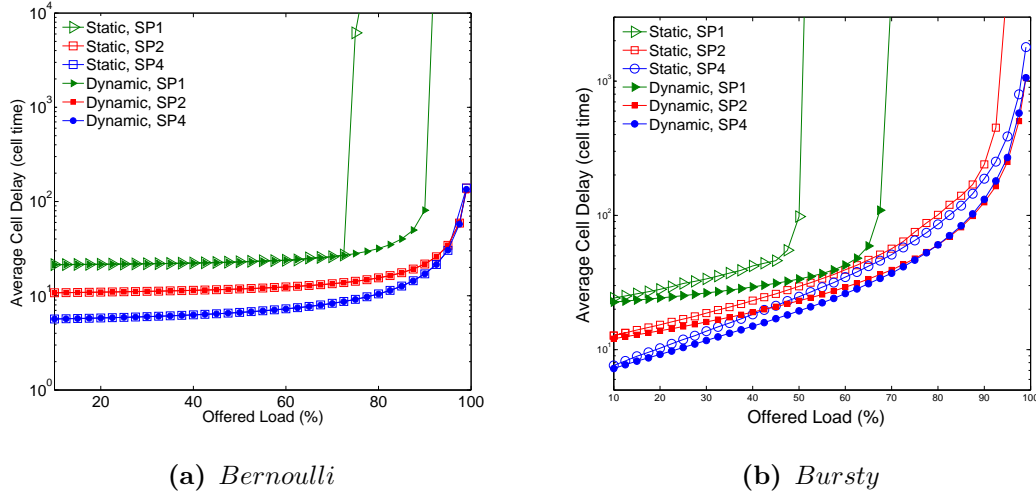


Figure 4.11: Delay performance of a 64-ports two-stage Clos-UDN switch under uniform traffic.

switch using a RR packet dispatching. Likewise, the performance of the two-stage switch with different SP values is inspected under the uniform bursty traffic. Figure 4.11(b) shows that the Clos-UDN switch with a static IM-CM connections configuration and a $SP = 2$ provides – nearly – 50% throughput. Setting SP to 4 makes the switch reach a delay performance that is a little bit higher than when a dynamic RR dispatching is put in use.

Unbalanced traffic

As mentioned in the previous chapter of this thesis, the unbalanced traffic is defined using an unbalanced coefficient ω that reflects the proportionality of the traffic distribution among the outputs. For an $N \times N$ switch, the traffic load from an input port s to an output port d is defined by $\rho_{s,d}$, where

$$\rho_{s,d} = \begin{cases} \rho(\omega + \frac{1-\omega}{N}) & \text{if } s = d \\ \rho \frac{1-\omega}{N} & \text{otherwise} \end{cases}$$

Figure 4.12(a) shows that using a speedup of one, the two-stage switch reaches up to 87% throughput. Making $SP \geq 2$ proves to be sufficient, in the sense that it makes the statically configured switch achieve full throughput. Note

4. A PACKET-SWITCHING ARCHITECTURE WITH UNI-DIRECTIONAL NOC FABRIC

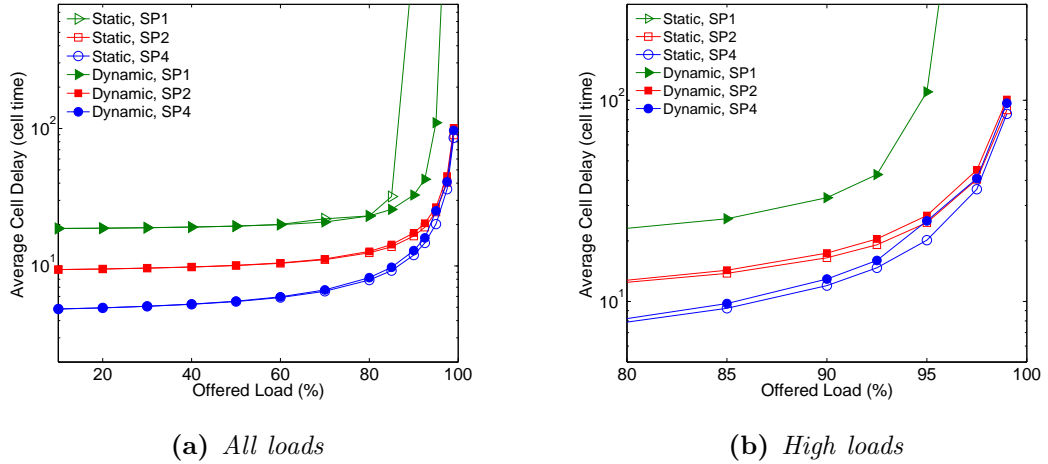
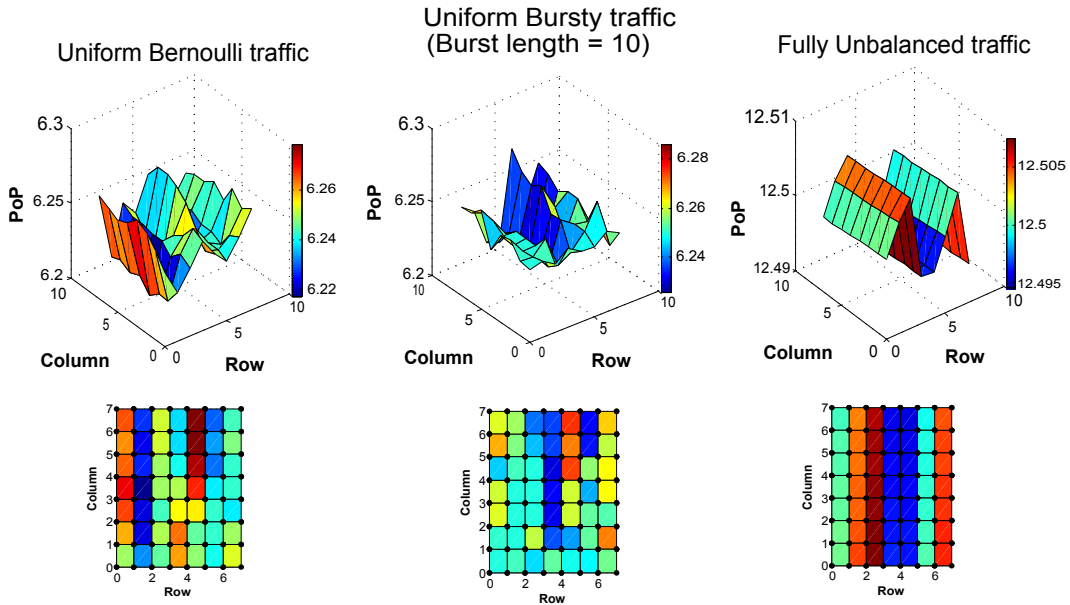


Figure 4.12: Delay performance of a 64-ports switch under Hot – Spot traffic.

that for high traffic loads, the average end-to-end packet delay of the two-stage Clos-UDN switch becomes slightly better than the three-stage Clos-UDN with dynamic packet dispatching, as depicted in Figure 4.12(b).

It is clear that the dynamic packet dispatching constantly disburses the traffic through the LI links which results in a good load balancing within the different UDN modules. On the contrary, the static scheme makes the load partition between CMs strongly dependent on the traffic type. If the switch is fed with skewed traffic, some LI links (and consequently UDNs) might be loaded. Packets exiting the Clos-UDN switch are intercepted and an analysis is made for the Proportion of Packets (PoP) going over the *East* links of the of the UDN’ mini-routers. Figure 4.13 illustrates the average PoP of the *West-East* traffic calculated over 8 UDNs (in a 64-ports Clos-UDN switch operating with UDN SP = 2). Clearly, packets get equally distributed among *East* links for both dispatching schemes under uniform traffic arrivals and that for a critical diagonal traffic (where input i of the switch sends traffic only to output of the same index), the dynamic RR dispatching contributes to a better load distribution in the UDN modules.

Dynamic dispatching scheme



Static dispatching scheme

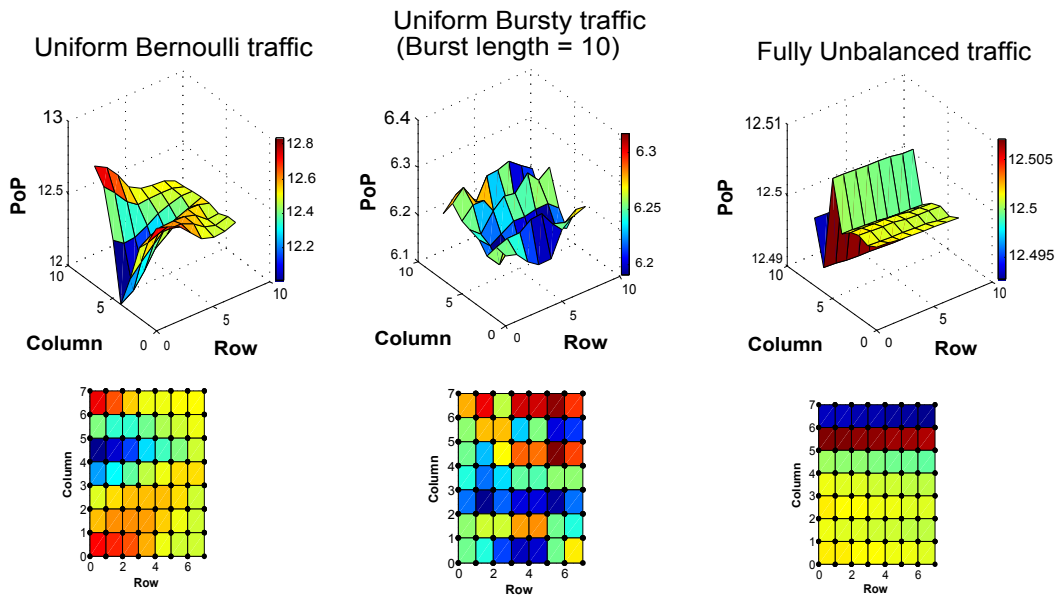


Figure 4.13: Proportion of Packets moving west-east in the middle-stage UDNs.

Throughput stability of the two-stage Clos-UDN switch

Using no speedup ($SP = 1$) limits the switch performance under a non-uniform traffic for both, a static and a dynamic dispatching methods. The UDN fabrics being tanked with heavy traffic loads, cannot afford full throughput if the on-chip links run at the same speed as the external line rate. However, slightly increasing the UDNs' speedup enhances the switch performance and a $SP = 2$ makes the two-stage Clos-UDN reach full throughput under the complete range of ω , as Figure 4.9 depicts.

4.6 Related work

In this part a brief chronological history of research on scalable and high-radix multistage packet-switching architectures is given. Because the body of the previous work is too large, this section is by no means exhaustive. The review is restricted to some of the key salient proposals.

The multistage connects were first proposed to interlink processors and memories in large multiprocessor systems that were the hardware foundation of fast supercomputers. The paradigm was later adopted to design scalable packet-switches that keep up with the growing networking needs. Large valency, higher throughput, lower packet delay and reliability are primary requirements of DCN backbone switches/routers. Many multistage interconnect patterns have featured the state-of-the-art work. Delta class topology such as Banyan, Omega, Baseline, and n-cube provide a unique path between each input/output tuple. They offer less path diversity than some other patterns. Yet, it happens that they also have simple self-routing propriety. The non-blocking Clos-network is another very popular design [61] generally quoted as $\zeta(m, n, k)$, where m , n , and k are the parameters that completely define the topology. Extensive work has been done on the Clos-network switches in all their variants such as S^3 [63], MSM [64, 69, 70], SMM [62] and MMM [74, 77] switches. Unfortunately, none of the existing Clos-network packet-switching architectures has been shown to simultaneously provide scalability in terms of cost, performance and hardware complexity. The MSM architecture proclaims expensive and complex input modules. Every

input module is required to cater for a high number of separate FIFO queues $(n.k)$ ¹ to avoid the HoL blocking. Besides, each of these queues needs run $(n + 1)$ times the external line rate [64]. Packet scheduling/dispatching is a big challenge in an MSM switch. Usually, a two step-scheduling process is required to resolve the input-output ports contention. In addition to its high cost and long configuration delays, no scheduling algorithm for this architecture has been shown to exhibit satisfactory performance [69, 70]. As for the MSM architecture, the MMM switch calls for N VOQs at the IMs to prevent HoL blocking. The buffered architecture [74, 77] mandates expensive internal memories to relax the scheduling process. Fully-buffered Clos architectures have good throughput performance since all contentions are absorbed by means of the internal buffers.

The Clos-UDN switching architecture differs from all aforementioned proposals in many ways. A radically different approach is taken to redesign the middle-stage of the Clos-network switch by adopting a NoC fabric. Designing each CM as a NoC tenders a number of advantages, and helps overcome previous limitations of conventional crossbars. First, input modules are less complex and cheaper compared to former architectures. Each input module of the Clos-UDN switch requires only m input FIFO queues, each of which runs *twice* the line rate. This is to be compared to the MSM and MMM switches, where each input module requires $(n.k)$ input FIFO queues each of which runs $(n + 1)$ times the line rate. Contrary to the complex, costly and under-performing proposed schedulers in traditional Clos architectures, the Clos-UDN uses a fully distributed and parallel scheduling mechanism, making it simple, fast and efficient.

The dynamic dispatching scheme disorders packets as it distributes them to different UDNs through time. By imposing a static configuration of the input and central modules links, the number of stages of the Clos-network gets reduced to two instead of three. Two-stage interconnects are more scalable than single stage architectures. However they are blocking if only one link

¹Provided a Clos-network of size N , where $N = n \times k$, the first stage would made of k input modules each of size $n \times m$. The middle stage would have m switches each with k inputs and k outputs. Last, there would be k output modules at the third stage, each of size $m \times n$.

is used to connect any input/output pairs of modules. Hence, connections must be built in redundancy and a large number of links between the first and second modules is required. The load balanced Birkhoff-Von Neumann switch made with two stages of crossbar switches was proposed in [163]. The switch is made of input-buffered modules in the second stage. It has no schedulers and adopts a deterministic sequence of N different configurations to connect N input/output pairs of modules. A disadvantage of the two-stage switch is that it can experience out-of-sequence packets delivery [162]. To prevent packets mis-sequencing and to maintain performance benefits of the two-stage switch, I. Keslassy *et al.* introduced expensive three dimensional queues (3DQs) and a frame-based scheduling algorithm [162]. In a different approach to build scalable switches, R. Rojas-Cessa *et al.* discussed a bufferless two-stage scalable switch with module-first matching scheme [164] where an iterative matching is performed between the input and output modules at the first place and ports matching occurs later.

Although they are interesting, two-stage interconnects have several limitations which urged other architectures to rise. As for the Clos-UDN switch, a static configuration for the IMs and CMs connections is imposed. Subsequently, packets of the same flow stored in the same input FIFO are constantly sent to the same UDN block, where they are routed to their corresponding outputs in the order of their arrivals. It was shown that a static configuration simplifies the switch and maintains good performance under a range of traffic types while preserving packets order. Furthermore, input schedulers are no more needed and the switch architecture can be viewed as a two-stage interconnect with in-order packets delivery guarantee.

4.7 Summary list

1. Single-stage crossbar switches are *absolutely* non-blocking. However, building a large crossbar-like switching fabric necessitates many crossbar chips. It is rather costly than unpractical, since the number of crosspoints is still quadratic to the switch valency, *i.e.*, $\mathcal{O}(N^2)$.

2. The Clos-network packet-switching fabric is one of the most representative and successful designs. It can be optimally non-blocking, or simply *rearrangeably* non-blocking. It offers good scalability features, and it can serve to build cost-effective high-radix switching architectures.
3. So far, many variants of Clos-network switching fabric architectures were proposed and studied. Unfortunately, almost all designs exhibit costly modules or highly-complex scheduling algorithms.
4. In general, conventional three-stage switches require VOQs to alleviate the HoL blocking. An N -ports switch needs a total of $(k \cdot N)$ FIFO queues (where k is the number of input modules), each running $(n + 1)$ times faster than the external line rate.
5. Bufferless Clos-packet switches typically need two steps to schedule packets. First, output port contention must be resolved for all input ports in a way similar to a traditional single-stage IQ crossbar switch. Second, the global path allocation must be performed to assign conflict-free routes for already matched pairs of input/output ports.
6. In buffered Clos-switches, the scheduling process is no more a challenging point. The contention is absorbed by internal queues that serve to temporarily handle contending packets. However, large memories might be needed to keep up with the traffic unbalance. The out-of-sequence packet delivery is another issue that is inherent of the buffered nature of the middle-stage switching modules.
7. In this chapter, a new way to design the heart of a multistage switch is proposed. Using a uni-directional Network-on-Chip fabric, a scalable switching architecture is built. Interestingly, the switch obviates the need for costly IMs, complex centralized schedulers, and expensive internal buffers.
8. Simple RR packet dispatching scheme serves to dynamically distribute packets among the central UDN modules, where they get routed hop-by-hop until the external LC links.

9. The dynamic packet dispatching disperses packets of the same flow among the CMs. It makes the switch achieve high performance, but it also results in a disordered packet delivery.
10. A static configuration of the IM-CM interlinks is proposed. The connection pattern limits the switching facility, and reduces the three-stage Clos-network switch to two-stage switch. Simulations show that, most of the time, the performance is marginally affected. All the same, packet flows are firmly dispatched to the same CM. They are routed across the NoC-based fabric through deterministic paths, which results into an ordered packet transfer.

4.8 Conclusions

In this chapter, the Clos-UDN switch is presented. It is about a scalable and easily configurable multistage switch with simple FIFO queuing at the input modules and simple packet dispatching schemes. NoC-based fabric modules are implemented with on-chip buffering and arbitration in the middle stage of the Clos-network. The NoC fabric allows a pipelined and distributed scheduling, and obviates the need for a centralized and complex arbiter. The current design also avoids the use of large crosspoint buffers like those that fully buffered structures require (the MMM switch). Adopting a dynamic dispatching process mis-sequences packets delivery in the same way MMM does. It is interesting that packets can be prohibited from getting dis-ordered at first place, by introducing a static configuration of the IM and CM modules interconnections. Although it reduces the switch architecture to two-stage Clos-network, this approach results in constantly dispatching packets of a given flow to the same CMs where they get forwarded in-order and in a multi-hop way until their output ports using deterministic routes. Intensive simulations have shown that the three-stage Clos-UDN provides high and stable throughput. Using a minimum on-chip speedup factor, $SP = 2$, the Clos-UDN switch gives good average latency as compared to MSM and MMM switches under a range of traffic patterns for far less complex architecture and scheduling process. Overall, the Clos-UDN switch demonstrates:

- A robustness of packet delay to the switch valency
- An immunity of overall delay in presence of hot-spots
- Almost constant delay variation under medium-to-high loads no matter the switch size and the traffic type are
- High achievable throughput

Based on the previous works and the current technology advances, running the switch central modules (UDNs) with speedup of two is said to be quite straightforward.

5

A Congestion-Aware Clos-UDN switch

5.1 Introduction

In Chapter 4, the Clos-UDN switch is described and its performance is evaluated. While the packet-switching architecture is interesting, it relies on a congestion-oblivious packet routing algorithm. In reality, DCN switches deal with a huge volume of inter-server traffic. They are often subject to events of congestion that severely affect their efficiency. Although there is tremendous interest in designing improved switching architectures for DCNs, few proposals suggest solutions to amend the congestion management at a switch level rather than the DC network level [165]. In the context of data centers, managing the constantly increasing loads is pivotal. The move of load-balancing functionality in DC networks has been motivated by the apparent necessity of having a global load and congestion administration in the switching substrate. Some of the latest papers struggled to convey the load-balancing function to centralized controllers [166], the network edge [165, 167] or end-hosts [168]. At first glance, a global load-balancing seems sufficient to manage the load distribution across the DCN. However, real-world measurements reveal that congestion events are mostly short-lived. They are too brief that a global synchronization loop can simply miss them. In this chapter, an enhanced version of the Clos-UDN is presented. The switch presents neat features, and adopts a congestion-aware packet routing to better respond to the traffic non-uniformity.

5.2 Congestion-aware load-balancing in DCNs

DCNs support a large array of applications that provide different workloads ranging from latency-sensitive small flows to bandwidth hungry large ones. The variability of the traffic loads imposes the need for load-balancing in order to alleviate (or correct) congestion events. Load-balancing is crucial to preserve the operational efficiency of the network, and to maintain suitable application performance. Figure 5.1 gives some examples for centralized and distributed load-balancing techniques in DCNs cited in the literature. Commonly, load-balancing in DCNs is performed using a centralized engineering mechanism [8, 53, 166], multipath transport, or specialized hardware. The centralized congestion and load management have been motivated by the perceived necessity of collecting a global congestion information to evenly distribute the load among potential paths. The centralized approach operates at coarse time scale, and proved to be too slow for volatile traffic of the DCN. More importantly, deploying a central arbiter in large-scale networks with high transmission rates is challenging. It is subject to scalability limitations and most often, it provides limited performance [169]. In stark contrast to the centralized solutions, distributed traffic engineering have been suggested to mitigate the scalability impediment, and to handle volatility in the DCNs. CONGA [165] is a mechanism that gathers and analyses congestion feedback from leaf switches located at the network edge. Expeditus is another distributed congestion-aware load-balancing scheme that explores path diversity to prevent mice flows from being routed through congested paths. Although they respond faster to congestion, distributed approaches have a high barrier to deployment [168] which largely comes from the specialized networking hardware needed to be in place. PRESTO [168] is a soft-edge load-balancing scheme implemented in the vSwitch, or hypervisor, to proactively distribute sub-flows¹ near-uniformly in the network. However, the approach is congestion-oblivious, and simply incurs a performance collapse in case of link failure. Based on the observation that the short-lived congestion caused by micro bursts holds for a large amount of packet loss, DRILL was proposed in [169] to introduce

¹Presto divides flows into equal size *flow cells*, each of size 64KB. Sub-flows are then sprayed in a RR fashion among available paths.

the micro load-balancing in DCNs. In contrast to the pervasive load-balancing methods which often rely on the global network status, DRILL enables finer load-balancing decisions using the local information readily available to each switch in the network.

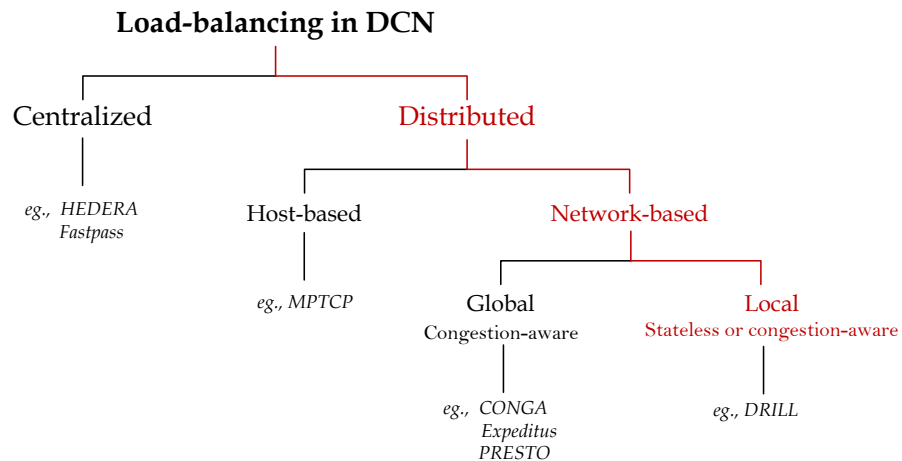


Figure 5.1: *Design space of the load-balancing techniques in DCNs [165].*

5.2.1 Micro load-balancing

Congestion and a bad load distribution, both, are the culprit of low latency in the DCN. While many schemes have been proposed to keep the latency range always minimal by monitoring queues [99, 170], others relied on priority and minimal buffering [100]. Yet, a good number of schemes steadily use the global network status to distribute, and to re-route the traffic flows. The current design trends aim at building a smarter switching fabric that significantly lifts up the overall performance. This work (in addition to others) makes a strong case for moving load-balancing in the data center switching substrate from the centralized controllers and the network edges, to the very basic unit: Switches. The micro load-balancing was first proposed to perform finer load distribution and to get rid of the slow and restrictive control loops of global schemes. It is

5.3 Description of the switch architecture

known that data centers experience congestion events that are in the majority short-lived. These incidents are due to brief spurts of high demand from some applications [171, 172]. They typically last for few microseconds while a global congestion information might take several RTTs (tens of microseconds) to collect and correct congestion. By that time, congestion is more likely to be over [171]. In [169], authors suggested DRILL which is a scheduling algorithm purely local to switches. The idea is to allow switches of the DCN to instantly react to load variations, and to redirect packets so that no queues build up fast and infect the throughput. A proactive approach with no need for any congestion feedback is adopted. In the following parts of this chapter, the switch architecture is presented, and the process of packet routing is explained.

5.3 Description of the switch architecture

Because of the limitation of the traditional Clos topology, additional alternative routing resources can provide more network tolerance and further improve the switch performance. This motivated the following idea:

***Idea.** *“Adding links in between the central modules would help build in more connectivity on the baseline Clos-UDN switching architecture that is described in Chapter 4. It become possible to reduce the traffic congestion in the whole Clos-network switch under critical traffic patterns by actively distributing the traffic load among the multistage switch modules”.*

Links in between the central modules are added to build in more connectivity on the baseline Clos-UDN switching architecture that is introduced in Chapter 4. As it shall be demonstrated later, inter-CMs links reduce the traffic congestion in the whole Clos-network switch under critical traffic patterns, and contribute to better load distribution among CMs. In what follows, the switching architecture is described.

5.3.1 Cross-module interconnection

The major concern is to design a scalable and easily configurable switch that meets high performance requirements of today and the next-generation DCN fabrics. For simplicity, the *Benes*' lowest-cost practical non-blocking architecture¹ is considered. The first stage of the switch comprises k IMs, each of which is of size $n \times m$. The middle stage is made of m UDNs, each of dimension $k \times k$. The third stage consists of k OMs, each of which is of size $m \times n$. A total of m FIFOs per IM are maintained, each of which is associated to one of the m output links denoted as $LI(i, r)$. Since $m = n$, each $FIFO(i, r)$ would be

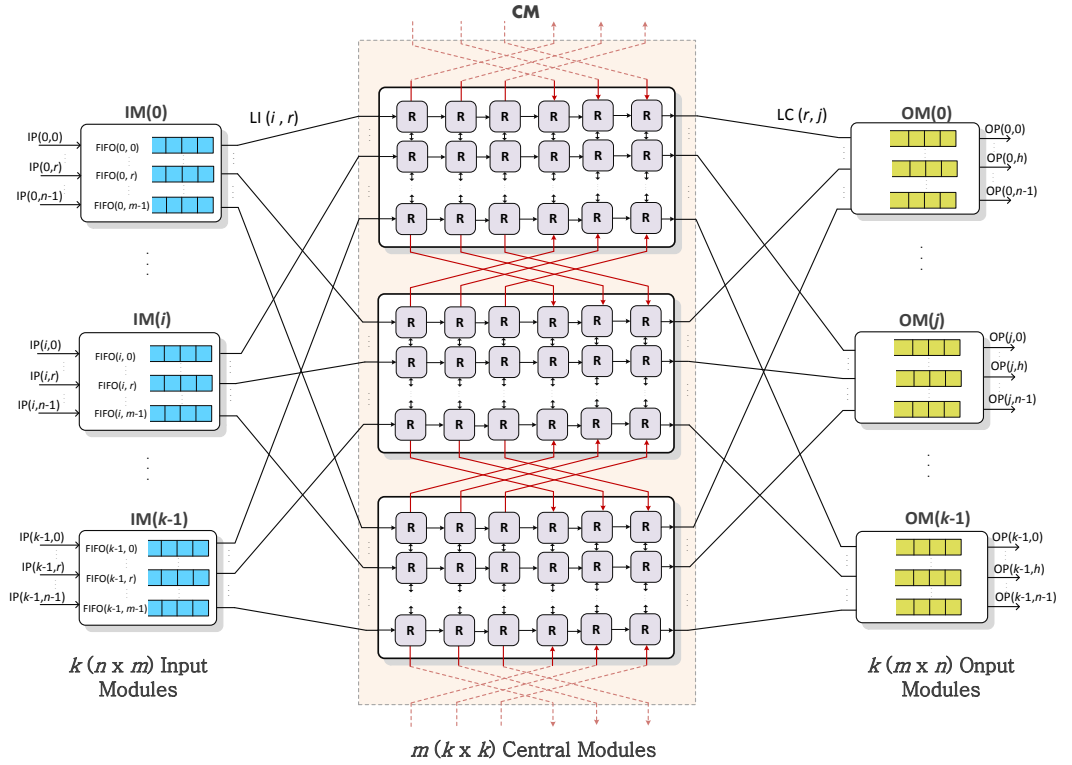


Figure 5.2: $N \times N$ Three-stage Clos switch architecture with cross-connected CMs.

also associated to one input port, $IP(i, r)$. It can receive at most one packet

¹Details about the dimensioning of the different modules in a generic Clos-UDN switch are provided in Chapter 4.

and send at most one packet to one central module at every time-slot. CMs are related to OMs with m links that are called $LC(r, j)$. An $OM(j)$ has n OPs, to which are associated n output buffers. An output buffer can receive at most m packets and forward one packet to the output line at every time-slot. Packets in the Clos-UDN switch are routed minimally using the “*Modulo XY*” algorithm. Traffic flows travel *West/East*, *West/North*, *West/South*, *North/South* and *South/North*. The previous results in Chapter 4 showed that a static packet dispatching, and an oblivious routing scheme are irrelevant to skewed traffic. Some UDN modules can get more congested than others resulting in longer delays and poor throughput. An elegant way is used to make the central modules of the switch share traffic and allow a proper load distribution. Taking advantage of the NoC design, a wrapped-around Clos-network is provided such that $CM(r)$ connects to $CM((r - 1) \bmod m)$ and $CM((r + 1) \bmod m)$ by means M unidirectional links. A total of $\frac{M}{2}$ links are used to send traffic to the upper (or lower – respectively) CM neighbour, and exactly the same number of links is used to receive traffic from an adjoining module as illustrated in Figure 5.2. Note that since links are unidirectional, no deadlocks can occur. Next, an appropriate routing algorithm is implemented to adaptively distribute the traffic load among the switch blocks. The next part provides details of the routing process.

5.4 Congestion-aware routing

Much research has gone into designing routing algorithms with provable behaviour. While these proposals typically assume a healthy network and a fairly distributed load, DCN switches frequently have non-uniform (and sometimes bursty) injection rates, and time-varying communications. This often leads to a temporary congestion (known as hot-spots). Broadly speaking, packet routing algorithms that have some flexibility with respect to the route choice within the switching fabric, yield multiple advantages over oblivious approaches. The latter are not able to adapt to the communication pattern and to the network status. The Congestion-Aware Clos-UDN architecture would be called the CA Clos-UDN switch, hereafter. A static dispatching process is implemented and very input

FIFO in an IM – persistently – sends packets to the same CM. There is no much intelligence in dispatching packets from the input stage to the NoC fabric in the central stage. Yet, the packet scheduling is reserved for the NoC modules at the heart of the Clos-network. Apart from the architectural dissimilarity, the routing algorithm is key difference between the current design and the baseline Clos-UDN switch. It is important to note that NoC-based switches are delay sensitive and that the overall design performance heavily depends on the nature of the routing scheme [173]. Therefore, functionalities are added to the minimal-routing “*Modulo XY*” algorithm rather than using a fully adaptive method that causes longer latency. In the default case scenario, a packet is minimally routed through the CM where it was first injected. When the original CM becomes congested, a packet can leave it to the nearest less-congested CM module. The routing decision takes into account the congestion estimate at different points in the Clos-network and forwards packets correspondingly. In the following section is explained how the congestion is evaluated through time.

5.4.1 Congestion evaluation

A combined metric is adopted to suit for the routing scheme and to correlate well with the global Clos-network congestion while being inexpensive to compute. The Regional Congestion Awareness (RCA) [173] approach is also adopted to evaluate and to propagate the congestion information across a UDN module, and its direct neighbour CMs (blocks of indices $((r+1) \bmod m)$ and $((r+1) \bmod m)$). Thanks to RCA, the locally competed congestion levels are compared to those propagated from a neighbour CM before taking the routing decision. At the level of a single NoC module, a routing quadrant is defined to be the sub-network limited by the packet’s current position in the 2D mesh, and the egress port through which it exits the current CM to the third stage of the Clos switch. The local CM information is defined to be the one readily available at a given CM module. This information represents the status of all nodes (also called mini-routers) that figure in the routing quadrant. Given its current position, a packet can travel in one of the two quadrants *North/East* and *South/East* with each quadrant having exactly two possible output directions excluding the

local port. Buffers occupancy is a classic congestion metric that reflects the load distribution in points of the network. To keep on routing packets adaptively through minimal paths, two metrics are combined: The buffers occupancy and the hop count.



Regional Congestion Awareness routing policy in NoCs

Load-balancing is a paramount concern in any network, no matter what routing scheme is in use. Much research has gone into designing reliable NoC routing with provable performance. Oblivious routing algorithms are not only the simplest, but also the most bounded. They are inherently unable to balance the traffic load as they are unaware of the network congestion status. Schemes that have a degree of flexibility with respect to route choice, outperform oblivious solutions in several aspects. Adaptive routing mechanisms tackle the congestion avoidance problem. They are omnipresent in commercial multiprocessors and others chips. Even an adaptive routing can be congestion-oblivious or congestion-aware. The latter considers local, regional or global congestion status to take the routing decision as shown in Figure 5.3.

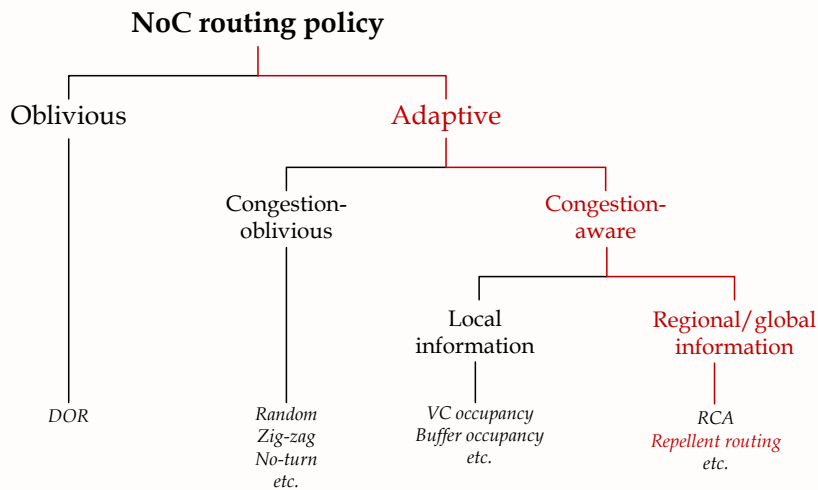


Figure 5.3: Design space for NoC routing policies with respect to congestion avoidance.

The RCA was proposed in [173] to consolidate the adaptive routing with

5. A CONGESTION-AWARE CLOS-UDN SWITCH

a congestion alleviation scheme in an on-chip network. RCA gathers regional information to improve the dynamic load-balancing across the network where packets are minimally routed. RCA predicts bottleneck paths by sharing the congestion status of every on-chip routers with its neighbouring routers. RCA calls for a mechanism to mix local (to the a particular on-chip router) and regional congestion information (nearby routers), which implies a higher hardware overhead. Yet, this technique provides better insight of the downstream routers for safe proactive packet forwarding. Subsequent research* undertook the design of lower cost congestion-aware algorithms with preserved reliability and performance [174, 175].

*This work does not try to cover all previous work in this part, as there has been intensive research on designing congestion-aware, low-cost, and reliable routing methods for NoCs. Only pointers are given to redirect the reader to useful related work.

5.4.2 The repellent routing

For NoC-based switches, adaptive routing algorithms are better than oblivious schemes whenever the traffic is non-uniform. Still, the adaptive methods can disrupt load balance due to local decisions that lack knowledge of the network status beyond the nearest neighbours of a node. In case of a 2D mesh network, adaptive routing algorithms congest the middle of the NoC and steer the traffic towards the center, leaving the edge nodes/links underutilized. The idea of the algorithm is to route packets across the CA Clos-UDN switch is as follows:

***Idea.** *“Modify the routing policy to make it possible for packets to exit a currently congested CM towards a less crowded module. This scheme is called repellent routing, as it tends to push a portion of the traffic to borders of the NoC module, as shown in Figure 5.4”.*

At every time-slot and at any position in a UDN module, a packet is subject to two levels of decision making: First, select the closest CM neighbour. Next, elect the less-congested routing quadrant. Considering the module $CM(r)$ of the CA Clos-UDN switch, the module $CM((r + 1) \bmod m)$ is said to be closer to

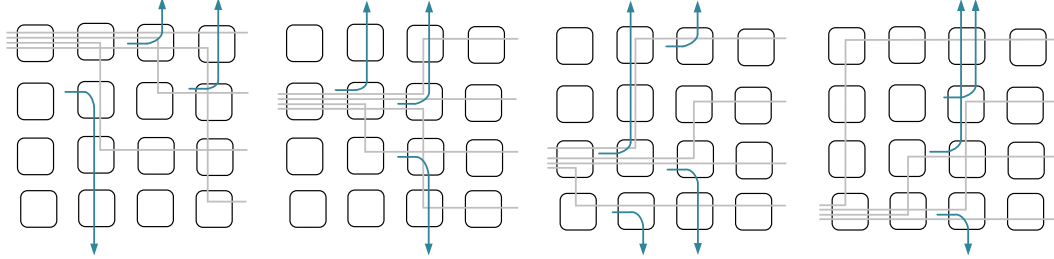


Figure 5.4: Example of the Repellent and “Modulo XY” routing algorithms.

$CM(r)$ than the module $CM((r - 1) \bmod m)$, if the vertical distance from the current node to the first row of the mesh, is less than that to the last row. As it was mentioned earlier, coupling the distance information with the status of the load distribution in the routing quadrant is relevant. It minimizes the impact of pushing packets back away from their destinations to be routed through another module. Whenever a cell is routed locally, it is the “Modulo XY” algorithm that would be in operation. Otherwise the packet is sent vertically *North* (or *South*) until the first (or last) row of the NoC where it leaves the CM to another block. Algorithm 2, gives details of the routing logic that the NoC-based modules adopt. It is worth saying that the crossed inter-CMs connections reduce the number of the NoC stages that a packet must go through until its corresponding LC link to avoid cumulating latencies and to prohibit the declining of the switch performance. As Figure 5.2 shows, a packets that flies its original CM, is liable to cross less pipeline stages in the new CM until the corresponding LC link. This connection choice is made to keep the transmission delays minimal.

Modifying the switching architecture and the packet routing process incites several changes. In the subsequent section, some practical considerations related to the CA Clos-UDN switch design are overviewed.

5. A CONGESTION-AWARE CLOS-UDN SWITCH

Algorithm 2: The repellent routing

```

1: if (pck_repulsed = TRUE) then
2:   port ← routing_direction
3: else
4:   pkt: choose closest CM
5:   if (local routing quadrant is less congested) then
6:     “Modulo XY”,
7:     pck_repulsed ← FALSE
8:   else
9:     if (chosen CM is UP) then
10:      routing_direction ← North,
11:      port ← North,
12:      pck_repulsed ← TRUE, //Override bit
13:     else
14:      routing_direction ← South,
15:      port ← South,
16:      pck_repulsed ← TRUE

```

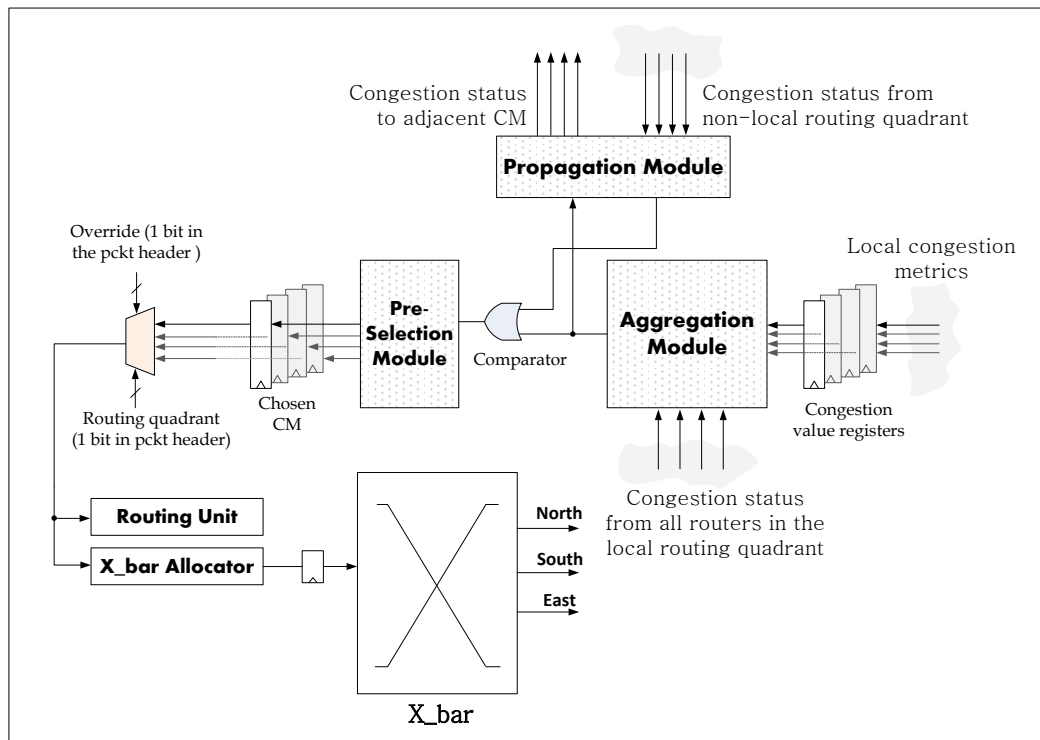


Figure 5.5: High-level diagram of an adaptive on-chip router.

5.5 Implementation issues

In this section, some of the issues that are related to the implementation of the congestion-aware switching architecture are briefly discussed.

5.5.1 The architecture of the on-chip routers

On-chip routers use the distance and the buffers occupancy of the downstream nodes within a routing area to evaluate congestion. In conventional adaptive routers, only intrinsic congestion information is used to select a preferred port (occupancy of output buffers as a common example). The RCA approach helps aggregating local¹ and non-local information to better estimate the congestion status [173]. Figure 5.5 depicts a high-level design of a mini-router. The aggregation module use the specified congestion metrics to combine values. Next, it feeds the result to both, the comparator and the propagation module. In contrast with the typical RCA mini-router, the current mini-router design uses the remote congestion information collected from the neighbouring CMs to make the routing decision. The pre-selection module keeps reference to the modules $CM(r)$, $CM((r - 1) \bmod m)$ and $CM((r + 1) \bmod m)$. Based on the output of the comparator, packets might be routed locally or sent to an adjacent bloc. The propagation unit transfers the congestion information from (and to) other nodes in the routing quadrant. Note that unlike the common RCA router – that sends information in a single direction – the current propagation module requires additional logic to convey the congestion estimate to nodes in a remote CM.

Recall that in the default case scenario, a packet is routed normally using the “*Modulo XY*” algorithm until the routing unit indicates that it should go through another less-congested CM. In which case, the packet will have to go the way *North* or *South* to exit the current UDN. Consequently, it must be able to override the value indicating the routing CM. This action is accomplished via an override bit in the packet’s header. In addition to the micro-architecture

¹Local to an on-chip node – mini-router.

of the on-chip routers, the in-order packet delivery is another challenge raised by the CA Clos-UDN switch design.

5.5.2 Re-ordering packets

Out-of sequence packets delivery is a common problem to packet switches with buffered fabric architectures. A re-sequencing mechanism at the output stage of the switch [77] is a popular solution to this problem. In an extension of the previous work [176], a static cell dispatching is employed to prevent the out-of-order packets delivery. However, the current proposal breaks this asset. Actually, sending packets across the middle stage of the Clos-network in a flexible way to better equalize the load, and to mitigate congestion mis-sequences the packets order. One of the previous ways might be used to resolve this issue. For instance, in [77], authors discussed two re-ordering schemes based on time-stamp monitoring. Although both alternatives do not require synchronization among the different SEs, many buffers and arbiters need to be introduced making these solutions unscalable. In [86], H. J. Chao *et al.* proposed several re-sequencing mechanisms such as: The static and dynamic hashing, and window-based resequencing. The switch requires a re-sequencing stage to re-establish the correct packets order. A better suited alternative that – mainly – does not defeat the cost and complexity purposes is proposed. In the following section, the performance of the CA Clos-UDN switch is assessed and compared to the state-of-the-art multistage switches.

5.6 Simulation results

Throughput and delay are the two most important performance metrics used to evaluate packet-switches and routers. In this section, the delay performance of the congestion-aware Clos-UDN switch is evaluated under different scenarios, and compared to that of the Clos-UDN with a static packet dispatching scheme. Also, a comparison between the performance of the CA Clos-UDN switch, the MSM [64] and the MMM [77] switches is given. Full mesh UDN central modules for which the depth $M = k$ are used in the simulations, if not explicitly

5.6 Simulation results

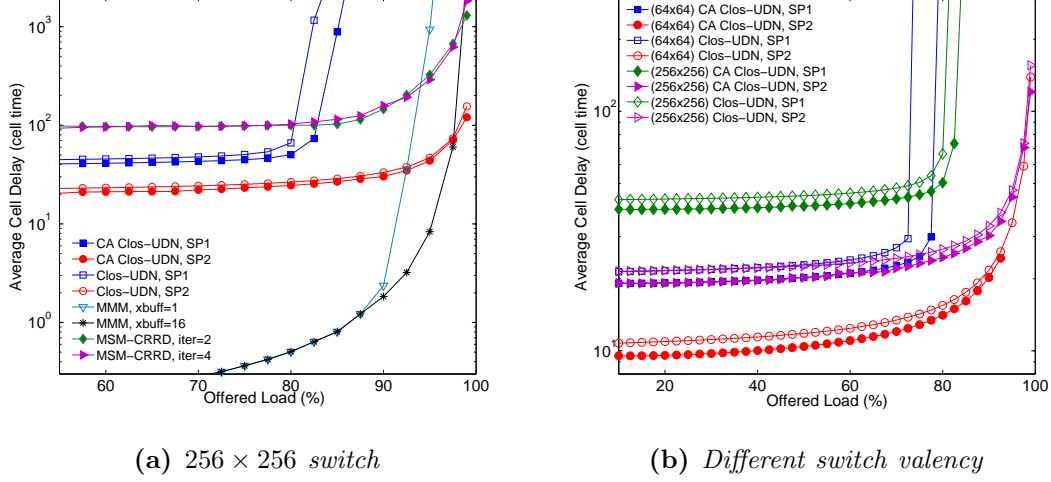


Figure 5.6: Delay performance in the Congestion-Aware Clos-UDN switch under Bernoulli uniform traffic.

mentioned. Remember that the plotted delay, is the averaged value over all packet queuing delays measured in a simulation.

The first test is performed for Bernoulli uniform traffic arrivals, for which results are shown in Figure 5.6(a). Simulations show that both the congestion-aware switch and the Clos-UDN switch perform poorly under light loads. All the same, the focus will be mainly on heavy loads as they are more relevant to the context of data centers. With no speedup a congestion-aware architecture improves upon the Clos-UDN switch with a static packet dispatching. Thanks to the inter-CM connections and the *repellent* routing scheme, packets continue to be routed minimally across the Clos-network taking into consideration the congestion levels in the modules of the middle stage. Note that the average packet delay is slightly reduced and that the throughput of the switch is boosted. Under heavy loads, setting the speedup of the UDN units to two, makes the CA Clos-UDN design outperform the MSM (even if the CRRD matching is iterated four times) and the MMM switch, with crossbar buffers worth of one packet each. In Figure 5.6(b), the switch valency is varied. The experimental results reveal that a congestion-aware switching architecture ameliorates the overall packet delay and the throughput even if no speedup is used ($SP = 1$).

5. A CONGESTION-AWARE CLOS-UDN SWITCH

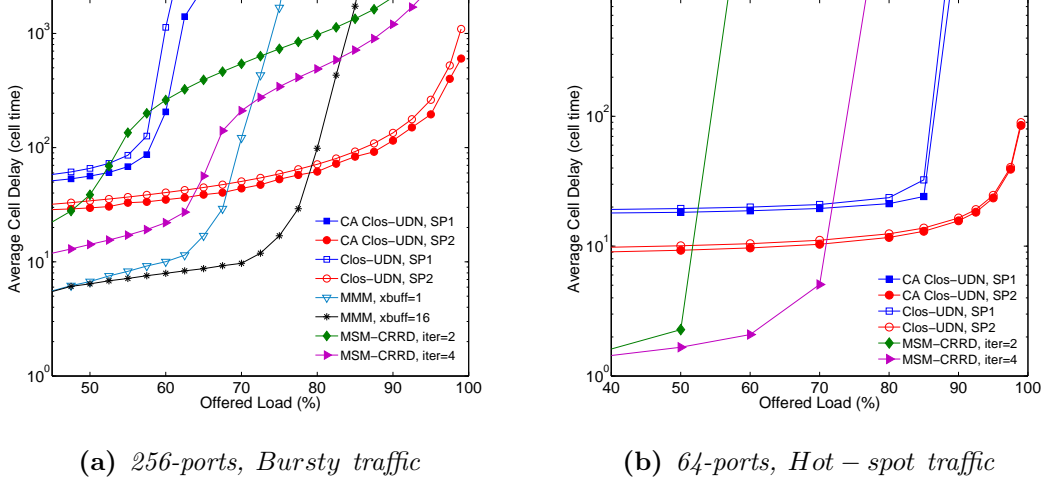


Figure 5.7: Delay performance under bursty and hot – spot traffic.

Workloads in the DCN are perpetually changing. Many high-bandwidth demanding applications make a bursty traffic relevant to DCNs with high-levels of peak utilization. In the simulations, the default burst length is set to 10 packets. A bunch of packets that arrive at the same on-period are destined to the same output port. As presented in Figure 5.7(a), the CA Clos-UDN switch decreases the end-to-end latency and slightly improves the throughput. Experimental results show that it is possible to improve the switch response to burstiness by speeding-up the UDN modules. Thus for $SP = 2$, the CA Clos-UDN switch beats the MSM and the MMM architectures under heavy traffic loads.

The uniform traffic is not realistic. Therefore, the next set of simulations are run for some unbalanced traffic patterns to test the current design’s robustness to non-uniformity. The following scenarios are considered: Bernoulli unbalanced, diagonal, and hot-spot arrivals. An unbalanced traffic model uses a probability, ω to disproportionate the distribution of the input load. A fraction would be sent to a predetermined output, while the rest of the input load would be uniformly directed to other output ports. As compared to the Clos-UDN switch with static dispatching scheme, the congestion-aware switch provides lower delays thanks to the adaptive routing that is used. More importantly, it outperforms

5.6 Simulation results

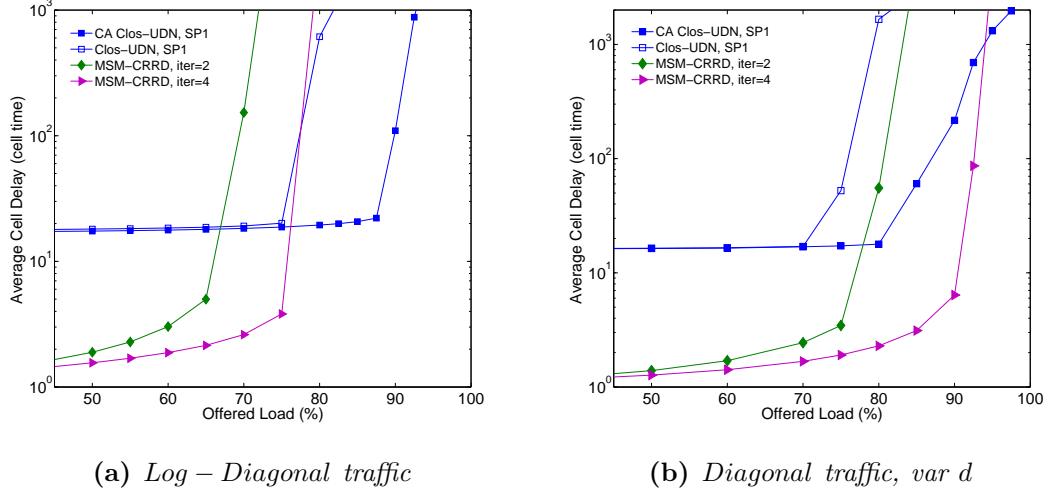


Figure 5.8: Delay performance of 64×64 switches under diagonal traffic.

the MSM switch with CRRD scheduling under medium and high loads even when $SP = 1$. More simulations are run while considering the minimum SP value and a switch size 64×64 under a diagonal traffic. A diagonal traffic can be represented as $d\rho(i, j) = d\rho_i$ for $i = j$ and $(1 - d)\rho_i$ for $((i + 1) \bmod N)$, where N is the generic switch size and ρ_i is the load at input i . Figure 5.8(b) and Figure 5.8(a) compare the delay performance of the congestion-aware switch to the Clos-UDN with a static packet dispatching and to the MSM switch. The CA Clos-UDN architecture used along with an appropriate routing is more effective under skewed traffic pattern. With no speedup, the CA Clos-UDN switch distributes better the load across the Clos-network modules, and achieves high throughput.

In Figure 5.9(a), the throughput of the different switches is compared under Bernoulli uniform traffic. Changing the coefficient ω from 0 to 0.5 corresponds to shifting from a uniform traffic to a hot-spot traffic. For $SP = 1$, a congestion-aware design increases the throughput of the baseline Clos-UDN switch. Additionally, it performs better than both MSM and MMM switches, under medium and heavy traffic loads. Increasing SP to two, makes the congestion-aware Clos-UDN switch insensitive to the traffic variation – as it unconditionally delivers full throughput. In Figure 5.9(b), the value of the

5. A CONGESTION-AWARE CLOS-UDN SWITCH

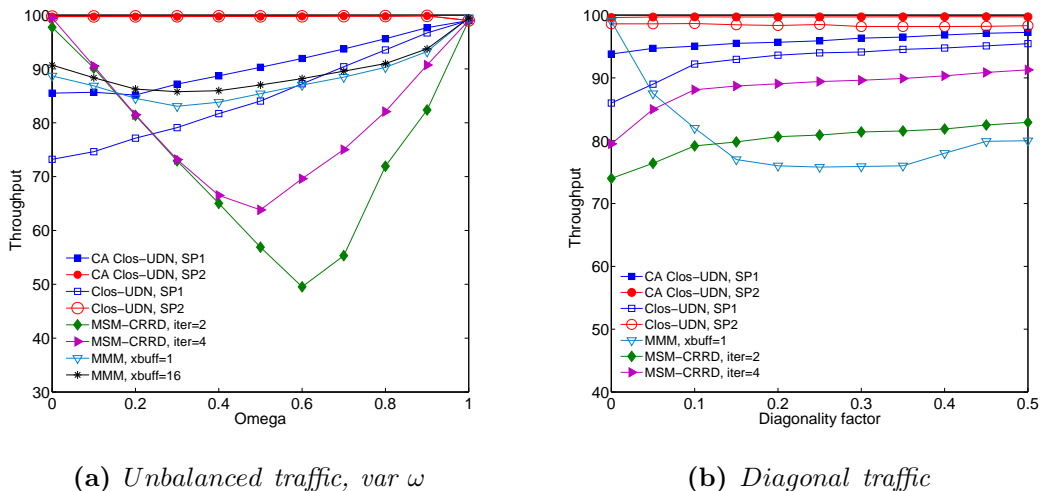


Figure 5.9: Throughput stability of different switches under non – uniform traffic.

diagonality coefficient d is altered to see how the throughput of different switches evolve. Simulations show that the response of the MSM and the MMM switches to diagonal traffic is poor under almost the whole range of d . On the contrary, the congestion-aware Clos-UDN multistage switch delivers up to 96% throughput assuming no speedup is used, and full throughput for all values of $SP \geq 2$.

5.7 Related work

One of the main concerns in data center switching fabrics, is to assure continuous load-balancing. This is a key point to enhance the network performance and to promote its robustness to floating traffic. Few multistage switch designs have considered load-balancing among the SEs using architectural and/or algorithmic solutions. The two-stage load balanced Birkhoff-Von Neumann switch was first introduced in [163] where the first stage balances the traffic load and the second stage performs the switching function. In [177], Smiljanic suggested some load-balancing algorithms for a three-stage Clos-network. In [178], Chrysos presented a distributed congestion management scheme for a buffered three-stage Clos switch and evaluated its performance under different traffic patterns. In a

more recent work, DRILL was suggested for single-stage switches¹, to perform a micro-load balancing within the DCN. The algorithm inspects the status of the output queues of the switch to help the input ports send packets to the less congested output ports. Interestingly, the scheduling logic is liable to apply to a multistage switching architecture. In previous works [176, 179], the classical Clos-network architecture is changed and UDN modules are plugged into the middle stage – instead of the common crossbar/memory blocks. Although it presents good scalability and flexibility features which translates into good performance, the Clos-UDN switch deals blindly with congestion spurts. This is likely to degrade the switch performance as the Dimension Order Routing (DOR) routing used for the Clos-UDN switch, is congestion-oblivious. On the contrary, adaptive routing algorithms [173, 180, 181] are a good alternative to improve the performance since they tolerate failure, and make intelligent arbitration based on the network status.

Under strongly unbalanced traffic patterns, the baseline Clos-UDN with no speedup ($SP = 1$) suffers bad load distribution which leads to local congestion, and affects the overall performance. The Clos-UDN architecture is modified by building a wrapped-around Clos-network. The switching facility is extended by means of interleaved CM connections. This contributes towards an increased path diversity. To actively balance the traffic load within the switch, a congestion-aware algorithm is implemented. Primary tests have shown that the CA Clos-UDN switch performs well under a range of traffic types. The evaluation of the CA Clos-UDN switch when integrated into a DC network is reserved for the future work. All the same, preliminary results attest of the use of the congestion-aware routing to promote the switch performance. This would also be of positive impact on the DCN performance.

5.8 Summary list

- 1- Data center networks are often subject to short-lived congestion events that rise in response to the high demand of some applications, which urge the

¹The algorithm was evaluated for single-stage $M \times N$ CIOQ switch in [169].

need for efficient load-balancing systems to resolve congestion and to preserve the network performance.

- 2- In addition to common solutions that suggest a global load management, recent works emphasized the need for fine-grain load-balancing to take place at the switch level. This is generally referred to as micro load-balancing.
- 3- A congestion-aware packet switch that accumulates advantages from the multistage design and the Networks-on-Chip paradigm is described.
- 4- The architecture relies on a wrapped-around three-stage Clos-network with NoC central modules, where interleaved inter-CM connections extend the switching facility, enhance the path diversity, and allow for load distribution throughout the multistage network.
- 5- An adequate routing process that combines the “*Modulo XY*” and the *repellent* routing algorithm is also proposed.
- 6- Packets are proactively routed using local congestion-status available in the current routing quadrant.
- 7- Aggregating a macro and a micro load-balancing techniques in the DCN is most effective to deal with any congestion event of any scale.

5.9 Conclusions

In this chapter a three-stage Clos switch with UDN central modules and inter-CM connections is described. An appropriate routing algorithm is introduced to allow a better load-balancing among the middle stage blocks. The regional congestion awareness is adopted, and the micro-architecture of the on-chip routers is modified to make them capable of evaluating and comparing congestion status at different points of the Clos-network before they issue a routing decision. For more effective routing, the minimal path and buffers occupancy metrics are combined to give estimate of the local and remote congestion in corresponding routing quadrants. The performance of the CA Clos-UDN switch is compared to

the Clos-UDN switch using a static dispatching scheme, the MSM switch using CRRD and the conventional MMM switch. The main focus in this chapter is put on the switch performance under high loads as they are more relevant to the context of DC networks. The simulation results show that CA Clos-UDN design delivers high throughput levels and bearable delays under a variety of traffic types. Yet, packets are likely to cross different CMs, which results in an out-of-order delivery, and urges the need for a re-sequencing stage to re-establish the correct packets' order.

6

A multistage switching fabric with Output-Queued NoC modules

6.1 Introduction

Classifying the switching architectures can be done in many ways. In addition to single and multistage grouping, a design can be described with regard to packet buffers placement. The IQ switches with FIFO input queues, have poor performance since contention degrades the switch performance in an unpredictable way. Although VOQs solve the HoL problem and improve the throughput, the implementation complexity/cost makes an IQ switch impractical, mainly for large port counts. Moving buffers to the output side of the switch increases the internal bandwidth. It becomes possible to simultaneously transfer multiple packets to the same output. Theoretically, this queuing scheme results in 100% throughput, and delays packets by a fixed amount of time. However, the current technology still cripples the OQ switching architectures design, since for a switch of size $N \times N$, memory banks should run $(N + 1)$ times faster than the external line rate¹. Meanwhile, small size OQ routers are feasible. In this chapter, a sophisticated switching architecture that combines features from the multistage, NoCs, and the OQ design is discussed. The design is proposed to defeat the limitations of classical packet-switches. The main idea is as follows:

¹To ensure the absorption of up to N contending packets that simultaneously request access to the output buffer.

***Idea.** “Unlike an off-chip OQ packet-switch, the memory speed is much less restrictive. Though embedded routers have few I/Os (3 as a maximum bound), and given the technological advance in the field of memory synthesis, OQ UDN modules should be merely implementable at affordable costs”.

It brings about a nested three-stage Clos packet-switch with FIFO queues at the input modules and a dynamic packet dispatching process. The conventional central stage crossbars are replaced with OQ NoC modules. In subsequent parts of this chapter, the switch performance is compared to some popular multistage designs using simulations. Also, an analytical model is proposed for the switch throughput, average packet latency and the blocking probability under *Bernoulli* uniform traffic. The model is supported with simulation results.

6.2 Clos-UDN switch with output-queued mini-routers

First, details of the switching architecture and the packet routing algorithm are given. Besides, the implementation feasibility of the OQ Clos-UDN switch is discussed.

6.2.1 Nomenclature of the switching architecture

The OQ Clos-UDN switch is a nested network with a three-stage Clos macro architecture where NoC-based modules are plugged into the central stage of the switching fabric as shown in Figure 6.1. The first stage of the switch is made of k IMs, each of which is of size $n \times m$. The middle stage consists of m output-queued UDN fabric modules of dimension $k \times M$, each. The third stage has k OMs, each of which is of size $m \times n$. As in previous chapters, an expansion factor $\frac{m}{n} = 1$ is considered, which makes the three-stage Clos-network a non-blocking *Benes* fabric interconnect. Remember that an $IM(i)$ has m FIFOs, each of which is associated to one of the m $LI(i, r)$ links. A link $LI(i, r)$ is related to a module $CM(r)$. Each $FIFO(i, r)$ of an input module, $IM(i)$, is

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

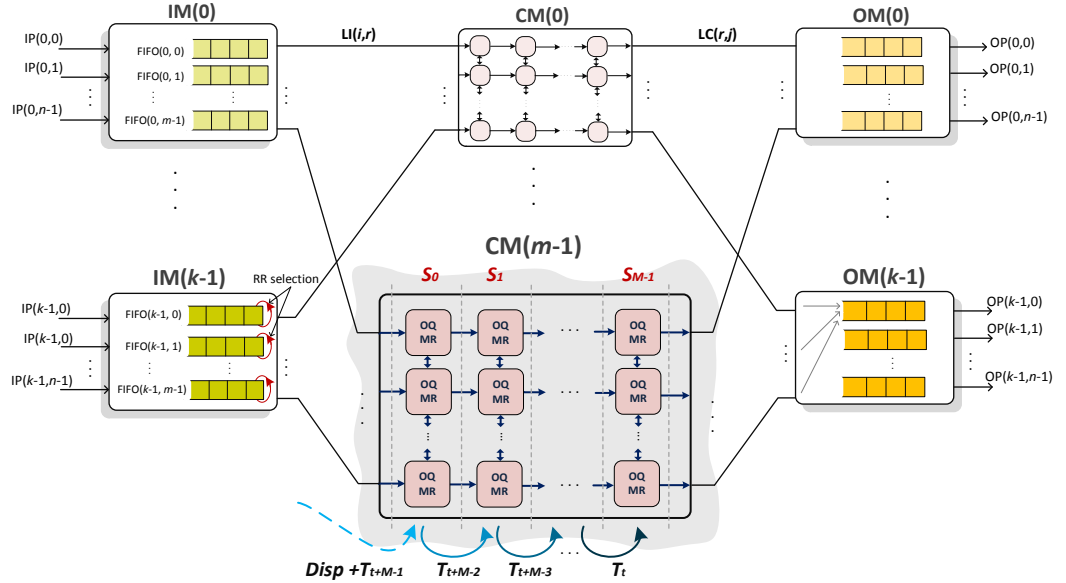


Figure 6.1: An $N \times N$ three-stage OQ Clos-UDN packet-switch architecture.

associated to one input port. It can receive at most one packet, and sends at most one packet to a CM at every time-slot. A CM has k LC links to connect it to the different OMs. An $OM(j)$ has n OPs, each of which is denoted as $OP(j, h)$ and has an associated output buffer. The output buffer receives at most m packets, and forwards one packet to the output line at a time.

Every switching element in the middle stage of the Clos-network architecture is fully defined by the 2-tuples (k, M) where k is the number of I/O ports and M is the depth of the 2D mesh (*i.e.*, the number of pipeline stages). A central module NoC assimilates $(k \cdot M)$ OQ mini-routers, each with two or three I/Os depending on its location on the grid. A deadlock-free NoC routing algorithm “Modulo XY” [151] and a credit-based flow-control mechanism are used to monitor the packets forwarding to the small on-chip buffers. This allows the upstream mini-routers to keep track of the free room in each output buffer downstream, and avoids elastic buffers.

6.2.2 Packet switching mode in CMs

As defined in Section 2.6.1 of Chapter 2, the switching strategy defines how packet flows cross the NoC fabric. When data traffic arrives to an input port of a source node s , the addressing information gets first checked. Upon making the routing decision, the adequate switching elements are made active and ready to move the user traffic to the next-hop. The time elapsed since then is collectively referred to as mini-router’s processing delay t_p . As some traffic might need to wait in line to be processed, packets might be stored in the output queues. This amount of time spent waiting there, is called the queuing delay t_q . Note that if multiple inputs of a mini-router receive traffic intended for the same port, then the output queue might be overwhelmed. And so, this is the reason why the store-and-forward switching mode is adopted to develop a backlog of frames waiting for the output port facility to become available. The *zero-load latency* of a network ($T_{network}$) is introduced to reveal how the physical structure of the network graph affects the central modules’ performance. Since the store-and-forward switching mode is in use, this performance metric can be expressed as follows:

$$T_{network} = N_{Hops} \times (t_p + t_{link} + L/bw) \quad (6.1)$$

With N_{Hops} being the average number of mini-routers that a packet has to traverse until the destination node d and t_{link} being the time to cross an on-chip link. The parameter L is the packet’s length (bits) and bw is the bit rate of the communication channel (intermediate link relating two adjacent mini-routers). In what follows, Is denotes an input port in a source node s and Od , an output port in a destination node d . The lattice distance between Is and Od is denoted $|s - d|$. Remember that the “*Modulo XY*” routing algorithm relies on the geometry of the grid and the destination of the packet to point out the packet’s next hop. At the end of every time step, the lattice distance between s and d is decremented by one, and the packet header information is updated. In the simplest case scenario where an independent and uniform selection of a source and destination nodes is in place, the routing algorithm would perform an average of $k/2$ vertical transmissions (since it selects the next

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

vertical hop from 0 to k with equal probability among all rows of the NoC mesh). Likewise, a packet has to cross M mini-routers horizontally to reach its destination node. Summing up the two quantities gives an average lattice distance of $N_{Hops} = (M + k/2)$.

6.2.3 Routing in the OQ-UDN modules

As in the IQ-UDN, packets are routed in an OQ-UDN module using the “*Modulo XY*” algorithm. This routing scheme is a simple, deadlock-free dimension-order algorithm that sends packets along one dimension, then along the second dimension of the 2D mesh layout. It also introduces an extra turn to better equalize the load distribution among available paths. Unlike crossbars where the routing decision is taken before sending the packet through the module, NoCs offer diverse set of paths, and incremental routing processes. The path computation is processed at every single node of the OQ-UDN fabric. As it was stated earlier, this removes the packet overhead that all-at-once routing algorithms create. A feedback-control signal is generated at each time a packet tries to access a saturated buffer. The whole routing process in the NoC central modules of the switch is made of two phases: Packets transmission and feedback-control, as illustrated¹ in Figure 6.2.

6.2.4 Implementation feasibility of the switch

Using NoCs to design packet-switching fabrics has been a gradual process, with interconnects evolving from a single bus to multiple buses with bridges and crossbars. The design itself is a key offering low communication latency, low power consumption and high modularity. It enables the switching fabric to handle various applications traffic with different characteristics. NoCs are inherently parallel. They allow a distributed arbitration for resources. Consequently, multiple transactions between the on-chip routers take place concurrently in different parts of the NoC layout. Design challenges include the implementation

¹In the Figure 6.2, the term λ_{I_x, O_y} is used to refer to the rate of traffic flowing from input I_a to output O_b of a mini-router. f_{O_y, I_x} denotes to the probability of the feedback-control issued by any output queue to any of the input ports.

6.2 Clos-UDN switch with output-queued mini-routers

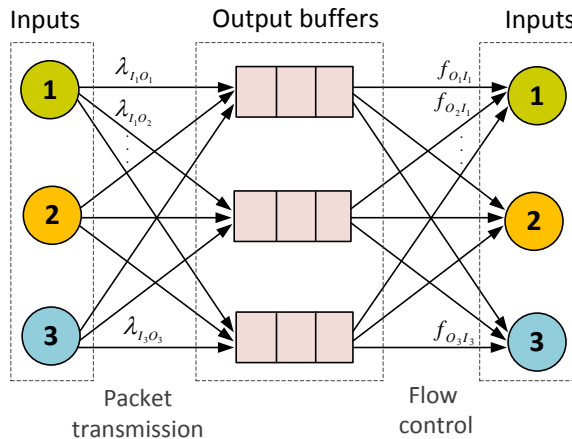


Figure 6.2: Routing process in an a mini-router of the OQ-UDN central module.

of the OQ Clos-UDN switch with competitive cost and feasible HW, while offering high-performance. As for the OQ Clos-UDN switch, two abstraction levels are required to pinpoint packet routes in the architecture. In this part, a rough estimation for the requirements of the multistage switch in general and the OQ-UDNs in particular is presented.

Once again, the dispatching of packets in the multistage switch is non-iterative in contrast with common semi-buffered Clos switches [64]. The latter requires maximum or maximal weight matching algorithms to define the set of interconnected IMs/CMs at every time epoch. Let's overview the packet scheduling process in the OQ Clos-UDN switch. At each time-slot, m RR input arbiters at the input stage select central modules and dispatch cells. This operation results in a complexity time of $\mathcal{O}(\log m)$ and also a hardware complexity of $\mathcal{O}(\log m)$ per IM. Every central bloc is made of $(k \cdot M)$ mini-routers, all fitted with small output queues that absorb traffic with respect to their capacity. For an mini-router of degree n , all output ports must run $n + 1$ times faster than an input port to handle the worst case scenario. Yet, the hardware implementation of the module is still feasible given the fact that the degree of a mini-router, n satisfies the following condition, $n \leq 3$ [182]. In the same work [182], authors presented a Register-Transfer Level (RTL) implementation of a single-stage Wrapped-around UDN (WUDN) packet-switch,

which fabric is quite similar to the OQ-UDN. Based on the technology advance that makes on-chip logic and memory VLSI implementation costs much less than off-chip communication, they show that the hardware of the WUDN switch is perfectly feasible.

Though it has a buffered middle stage, the OQ Clos-UDN switching architecture improves upon buffered Clos switches [75] since it has a more flexible architecture. It allows smooth change to the cost at the expense of a lower performance. As it shall be demonstrated, the OQ Clos-UDN switch provides good performance under a range of traffic.

6.3 Analytical modelling of the switch performance

After the design step, comes the evaluation of any packet-switching architecture. Usually, this is done through simulations. However, simulations are extremely slow for large-scale systems. They provide little insight on how the different design parameters affect the actual switch performance. Meanwhile, the analytical models, allow fast evaluation of large systems in early design phase taking advantage of the rapid trade-off design investigations. There is a great deal of interest in developing analytical models for the switching architectures as purely simulation analysis is not only inflexible, but also time consuming.

In this section, the performance metrics of the OQ Clos-UDN switch is analysed. But above all, the main focus is on the OQ-UDN modules which geometrical features differ from a simple crossbar. Some works that conferred modelling of NoCs and NoC-based switching fabrics are briefly reviewed. In 2009, Elmiligi *et al.* proposed an empirical model to address the queue size problem in OQ routers for NoCs using Markov chains analysis [183]. In a different approach, authors of [184] introduced a low complexity analytic method for the mean analysis of some performance metrics of NoCs. In 2010, Suboh *et al.* used a network calculus-based methodology to evaluate the latency, throughput and cost metrics of a NoC-based architecture [185]. In 2012, Fischer and Fettweis presented an accurate service estimation model for IQ NoC fabrics with RR

6.3 Analytical modelling of the switch performance

packet arbitration. Their approach is interesting as it takes into account the contention of multiple concurrent inputs jointly to the characteristics of the RR arbitration [186]. In [187], authors analysed the flow-control feedback probability between adjacent routers of a NoC as key step to evaluate the total performance of the network. An estimation of the average latency in a NoC was also provided in [188]. In 2015, Karadeniz *et al.* presented a low-complexity model for a wraparound single-stage switch based on Network-on-Chip and OQ routers [182].



Inside the OQ-UDN: Modelling the output-queues

The variety of parameters in the OQ-UDN modules is the essence for a high flexibility. It spans a large design space making room for both parametrization and optimization. Figure 6.3 depicts a high-level diagram of one mini-router of an OQ-UDN module. The output queues serve simultaneously as FIFOs, and can accommodate up to B^* packets, each. Every output has n input ports to serve ($n = 2$ or 3 depending on the on-chip router coordinates in the 2D mesh).

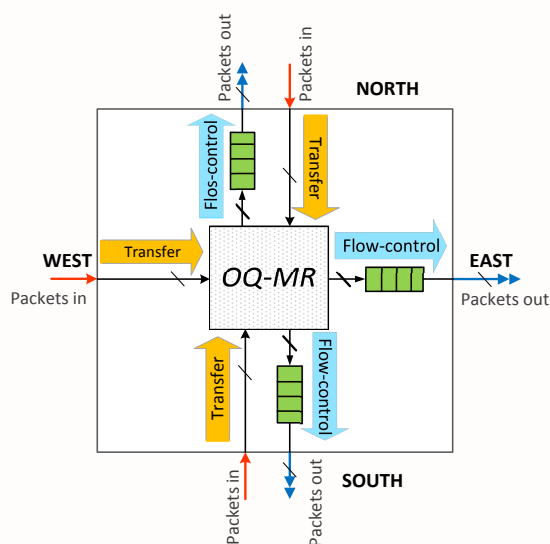


Figure 6.3: A block diagram of an Output-Queued mini-router.

It is common practice to study the finite capacity queues using Markov chains analysis. Assume a *Bernoulli i.i.d* packet arrival process with P_{arr} being the probability that a packet arrives to one of the outputs of a mini-router. The

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

parameter P_{dep} denotes the probability of packet departure from an output buffer. Also the arrival times and service times are assumed to be independent, and that they can – both – happen at the same time step. Finally, each output queue of a mini-router can be modelled as M/M/1/B queue which state transition diagram is shown in Figure 6.4. The transition probabilities of the buffer moving from one state to another are obtained by considering the ways in which a packet can move between the two states and the probabilities for these movements. Overall, the state transition in an output queue of the on-chip routers consists of two phases: First, verify the availability of the buffer space, and second move the packet forward by one NoC stage. For the M/M/1/B system, changes of the queue size occur by at most one per time step [183]. The term $b = 1 - p_{arr}$ denotes the probability that no packet arrives to the output buffer, and $d = 1 - P_{dep}$ the probability that a packet does not leave the output queue. To describe the transition diagram for the output queue, the following intermediate variables are defined.

- α : The probability that a packet arrives to the output buffer, but does not leave it at the current time step. This causes the number of packets in the output queue to increment by a unit.
- β : The probability that a previously arriving cell leaves the output queue. This decrements the number of queued cells in the buffer.
- f : The probability that the queue size remains intact. This can happen in one of two possible scenarios: A cell arrives at the output queue and another one leaves the same queue, or no cell arrives or gets removed from the queue at the current time step.

The state transition, diagram for the output queue is shown in Figure 6.4.

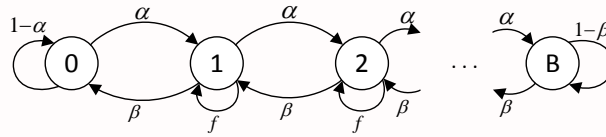


Figure 6.4: The state transition diagram for a single M/M/1/B queue.

It is possible to describe the system's state variation using the transition matrix of the output queue. The matrix can be written as:

6.3 Analytical modelling of the switch performance

$$P = \begin{bmatrix} \alpha_0 & \beta & 0 & \dots & 0 & 0 & 0 \\ \alpha & f & \beta & \dots & 0 & 0 & 0 \\ 0 & \alpha & f & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & f & \beta & 0 \\ 0 & 0 & 0 & \dots & \alpha & f & \beta \\ 0 & 0 & 0 & \dots & 0 & \alpha & \beta_0 \end{bmatrix}$$

where:

$$\alpha = p_{arr} d = p_{arr} (1 - P_{dep}) \quad (6.2)$$

$$\beta = (1 - p_{arr}) P_{dep} \quad (6.3)$$

$$\begin{aligned} f &= p_{arr} P_{dep} + b d \\ &= 2 p_{arr} P_{dep} + 1 - (P_{dep} + p_{arr}) \\ &= 1 - (\alpha + \beta) \end{aligned} \quad (6.4)$$

and $\alpha_0 = 1 - \alpha$ and $\beta_0 = 1 - \beta$.

Every element s_i of the state vector S indicates the probability of finding the queuing system in state s_i at that time step [189]. The first element s_0 reflects the probability that the queue is empty, while s_B is the probability that the queue is fully populated.

$$S = [s_0 \ s_1 \ s_2 \ \dots \ s_B]^t$$

The equilibrium condition of the output buffer can be written as: $PS = S$, which yields the following set of difference equations.

$$\begin{cases} \alpha s_0 - \beta s_1 = 0 \\ \alpha s_{i-1} - g s_i + \beta s_{i+1} = 0, \quad 0 < i < B \end{cases} \quad (6.5)$$

where $g = \alpha + \beta$. The system of equations in (6.5) is resolved, and the generic form of s_i is concluded

$$s_i = \left(\frac{\alpha}{\beta}\right)^i s_0, \quad 0 \leq i \leq B \quad (6.6)$$

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

The OQ state of occupancy changes over time. It can be one among the s_i states at a given time step which means that at a time, $\sum_{i=0}^B s_i = 1$. Thus, the probability s_0 that reports to the state of an empty queue is inferred.

$$s_0 = \frac{1 - \tau}{1 - \tau^{B+1}} \quad (6.7)$$

Where τ is the magnitude of the distribution vector S given by:

$$\tau = \frac{\alpha}{\beta} = \frac{P_{arr}(1 - P_{dep})}{P_{dep}(1 - P_{arr})} \quad (6.8)$$

Using the previous equations, the throughput of a single M/M/1/B queue is readily computed [190].

$$Th_0 = P_{arr} P_{dep} s_0 + \sum_{i=1}^B P_{dep} s_i \quad (6.9)$$

* B is the capacity of an on-chip output queue.

In addition to the inter-stage links, multiple routes are available within a single central module providing better path diversity and better load distribution. Starting their journey across the multistage switching fabric, packets need to be sent from the input module queues to the middle-stage OQ-UDNs. For simplicity and fairness, a RR selection of routes between any IM in the input stage and any CM in the central stage is used. The analytical approach of modelling and performance testing dictates knowledge about the parameters and inputs of the switch, to rigorously describe the architecture. In the following parts, it is assumed that on each input of the first stage, cells are generated according to an independent *Bernoulli* process. Also, the choice of output links $LI(i, r)$ at the packets dispatching phase is assumed to be independent and equidistributed [191]. Hence, the analysis can be broken to separately model the switching stages of the OQ Clos-UDN architecture. An approximated analytical model is built – mainly – by making use of the queuing theory and Markov chains. Remember that the set of eventual parameter values that impact the performance of the OQ Clos-UDN switch is very large. However, the OQ-UDN modules given their interesting proprieties. Next, the state transitions of the

6.3 Analytical modelling of the switch performance

OQ-UDN central modules are described, and the throughput of the multistage switch is computed.

6.3.1 Characterization of the throughput of the Clos-UDN switch

The throughput of the network is the rate of packets delivered to their ultimate destinations. At low traffic loads, the delivery rate is equal to the packet arrival rate, while it saturates with the increasing load [189]. The factors contributing to the throughput saturation are substantially the topology of the network, the routing algorithm, and the feedback-control mechanisms (if any is used). As for the current proposal, exits of the mini-routers (in the last column of the OQ-UDN modules) are related to output buffers in the OMs. For the sake of comparison with simulated switch performance, buffers of the OMs are considered to have infinite capacity, which means that analysing the throughput of the OQ Clos-UDN switch can be reduced to evaluating the packets delivery rate in the OQ-UDN central modules.

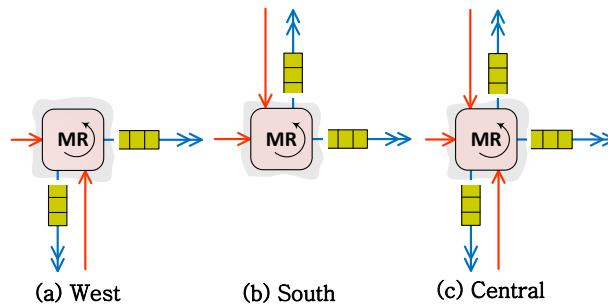


Figure 6.5: *Types of mini-routers in an OQ-UDN switch based on their degrees.*

An OQ-UDN module regroups three different types of on-chip routers based on the degree of the routers, as Figure 6.5 shows. Overall, there are $2M$ nodes of degree 2 and $M(k - 2)$ nodes of degree 3. As it was mentioned earlier, the central modules are supposed to work independently one from the other. Subsequently, to characterize the total throughput of the multistage switch, the

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

average number of packets that exit one central block is estimated. At this level, the packet arrivals to a given node on the NoC grid and the departure process are considered to be independent of each other. In other words, the average throughput of a single OQ-UDN module can be seen as the summed contributions of the last column's nodes. It can be described using the following equation

$$Th_{CM} = \frac{1}{k} \left(\sum^2 (Th_{deg_2}) + \sum^{k-2} (Th_{deg_3}) \right) \quad (6.10)$$

Where Th_{deg_2} and Th_{deg_3} are the throughput of mini-routers of degree 2 and degree 3, respectively. Since all I/O(s) of an on-chip router work independently and simultaneously to contribute to the average throughput of the node, the expressions of Th_{deg_2} and Th_{deg_3} using equation (6.9) can be derived as a summation of as many M/M/1/B queues as the degree of the mini-router.

6.3.2 Average end-to-end delay

In addition to throughput, the average packet latency is an important metric that helps evaluate the performance of a switching network, especially in multi-hop networks that are more delay-sensitive than single-stage point-to-point connected crossbars. The average end-to-end delay in the OQ Clos-UDN switch might be viewed as the contribution of, mainly, the input delay at the first stage of the Clos-network, and the delay across the OQ-UDN modules¹. Considering *Bernoulli* uniform packet arrivals to happen at the input ports of the OQ Clos-UDN switch, λ is defined to be the average arrival rate and μ to be the service rate. The mean waiting time in the input-stage of the Clos-network switch can be approximated using an M/M/1 system using the following equation

$$\bar{W}_{M/M/1} = \frac{\rho'}{\mu(1 - \rho')} \quad (6.11)$$

¹As it was mentioned earlier, to map with the simulation environment the output buffers associated to the switch output ports are considered of infinite size. Packets are considered to exit the OQ-UDN central modules they all leave the output buffers to their corresponding output ports after a fixed time.

6.3 Analytical modelling of the switch performance

where ρ' is the utilization factor equal to λ/μ' and μ' is the modified input FIFOs' service time subject to the node's buffers availability, P_{fwd} . The parameter P_{fwd} will be computed later.

$$\mu' = P_{fwd} \mu \quad (6.12)$$

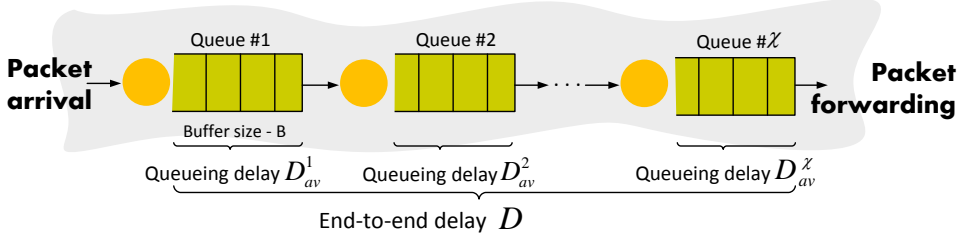


Figure 6.6: Packet delay in a tandem queue.

After being dispatched to the middle stage of the switch, packets cross the central NoC structures hop-by-hop, until reaching the LC links. A path (or route) is the set of successive links and output buffers that a packet has to cross from a source node to a destination node. A route is modelled as a tandem queue with χ output buffers as shown in Figure 6.6. Packets that are successfully received at the receiving side of each link are buffered in an output queue. They will be either transmitted to the next hop, or delivered to the third stage of the Clos switch. The delay of transmission over an intermediate link between two successive queues is assumed to be negligible in comparison to the buffering delay that depends on the following factors:

- The queue size B
- The probability that a queue has i packets, s_i
- The probability of packet arrival to the queue, p_{arr}
- The probability of service at the queue, P_{dep}

The average number of time steps that a packet spends inside the queue is given by:

$$D_{av} = \frac{Q_{av}}{Th_0} \quad (6.13)$$

where Q_{av} is the average queue size given by:

$$Q_{av} = \sum_{i=0}^B i.s_i = \frac{\tau[1 - (B + 1)\tau^B + B\tau^{B+1}]}{(1 - \tau)(1 - \tau^{B+1})} \quad (6.14)$$


and Th_0 is the throughput of the queue given by Equation (6.9). Using Little's formula for a tandem of queues [192], the end-to-end delay can be written as:

$$D = \sum_{j=0}^{\chi} D_j \quad (6.15)$$

where D_j is the average queuing delay at queue j and given in equation (6.13).

6.3.3 Blocking probability in the switch

In this section, the end-to-end blocking probability in the OQ Clos-UDN switching fabric is analysed. First, the term *blocking probability* is explained. Then, details of the analytical modelling are given.

 **The blocking probability** represents the probability that a finite-capacity buffer in a network is full. This implies that any arriving packet to this buffer would be lost. The blocking probability in a buffer at a time step t , depends on the initial queue size, as well as the arrival process. Obviously, high values of blocking probability have negative impact on the user's perceived performance of the network.

The on-chip output buffers have limited capacity. This means that it is necessary to control packet transfer to them. Under certain traffic patterns, packet flows invading output queues may rise the network's blocking attitude. Generally speaking, deriving the end-to-end blocking probability of a path in complex networks would be straightforward from the individual blocking probability of single links (unitary portion of the path), if the links are statistically independent. In case of the OQ-UDN switch, a path is made of passive input links (*i.e.*, that eventually impose no real constraints on packets transfer) and output queues of the on-grid routers. The availability of the buffering resource in outputs of the downstream mini-routers results in dependencies, and adds complexity to what could be a simple estimation of the end-to-end blocking

probability. Yet, it is possible to estimate an upper-bound on the probability that any path in the OQ-UDN switch is blocked.

The term P_{ctr} is the probability that an output queue issues a feedback-control signal at a time step. Referring to the previous analysis, s_B is the state where a single output port's buffer is fully occupied, which implies that

$$P_{ctr} = s_B = \tau^B \frac{1 - \tau}{1 - \tau^{B+1}} \quad (6.16)$$

Similar to the previous modelling approach, the OQ-UDN structure is unfolded. The end-to-end blocking probability of a route r in the OQ-UDN fabric is bounded by the sum of the blocking probabilities of its output queues. See Appendix B for the proof as follows

$$B(r) \leq \sum_{j=1}^{\chi} P_{ctr}(j) \quad (6.17)$$

Where χ is the number of buffers that a packet crosses in the NoC fabric. Next, computer simulations are carried out to evaluate the performance of the OQ Clos-UDN switch under different traffic scenarios.

6.4 Performance evaluation

In this section, the performance of the OQ Clos-UDN switching architecture is assessed under different traffic patterns using an event-driven simulator.

6.4.1 Uniform packet arrivals

The average end-to-end packet delay is assessed for different switch sizes, mesh depths M , and traffic types. Unless it is stated otherwise, the output buffers' capacity, B is set to the default value 3.

Uniform Bernoulli traffic

The delay performance of the OQ UDN design is tested when the fabric is part of a single-stage switch and a three-stage Clos switch under smooth traffic arrivals. In all figures, the notation OQ-UDN is used to refer to a single-stage switch,

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

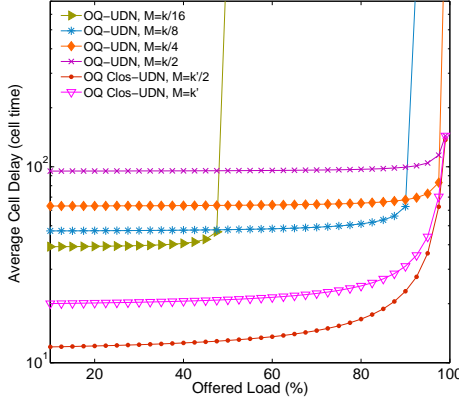


Figure 6.7: Delay performance of single-stage and multistage 64×64 switch, under Bernoulli uniform traffic, $B = 3$.

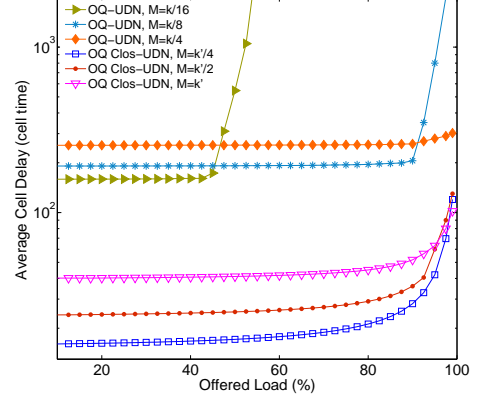


Figure 6.8: Delay performance of single-stage and multistage 256×256 switch, under Bernoulli uniform traffic, $B = 3$.

while OQ Clos-UDN is reserved for the multistage architecture. Figures 6.7 and 6.8 depict the variation of the delay time for a single and three-stage OQ-UDN switch under Bernoulli uniform arrivals for 64-ports and 256-ports switches. The parameters k and k' indicate the number of I/O ports of the standalone OQ-UDN and the Clos switch's central modules, respectively. Interestingly, whether used in a single or multistage architecture, the OQ-UDN design offers smooth delay variability for all proportions of the input load.

In the 64-ports switch, reducing M deteriorates the performance of a 64×64 single-stage switch. Unlikely, the OQ Clos-UDN switch seems less affected, as it keeps on delivering 99% throughput even with small M values. This mainly reports to what the multistage architecture brings over the single-stage design. Actually, breaking the unique large NoC into small units mounted in a Clos fashion reduces the size of the central modules. It becomes possible to distribute packet flows to various CMs where they are routed through smaller UDNs with much reduced congestion levels.

Note that at some point, reducing M leads to the saturation of the single-stage OQ-UDN, and that the multistage architecture offers better control on the absolute delay in large-scale switches as in the Figure 6.8 shows. Simulation results in Figure 6.8 clearly show that a single-stage design is unscalable to both port count and traffic load. A 256×256 single-stage OQ-UDN switch

can achieve full throughput only by expanding the NoC layout and setting $M = k/4 = 64$. However, this alternative is still unpractical and costly.

In Figure 6.9, the delay performance of the proposed Clos switch is compared to an MSM switch with the CRRD scheme [64], MMM [75] and the IQ Clos-UDN switching architecture [176] as being described in Chapter 4. The OQ Clos-UDN switch outperforms the MSM switch under heavy workloads where it categorically provides full throughput. The throughput of the IQ Clos-UDN switch saturates at around 90% provided that on-chip links run as fast as external LC/LI links (*i.e.*, $SP = 1$). An MMM architecture affords lower delay than all aforementioned switches. Yet, it still needs large crosspoint buffers to achieve full throughput (16 packets per crosspoint buffer). In the contrary, the OQ Clos-UDN switch running with small on-chip buffers ($B = 3$) and $M = k'/4$ (that is only equal to 4 for 256-ports switch port) ensures almost constant delay variations and provides high throughput.

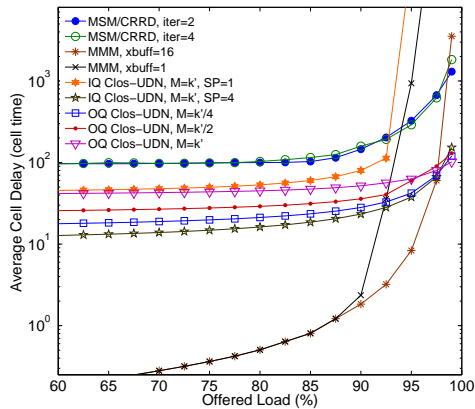


Figure 6.9: Delay performance of different 256×256 switches, under Bernoulli uniform traffic, $B = 3$.

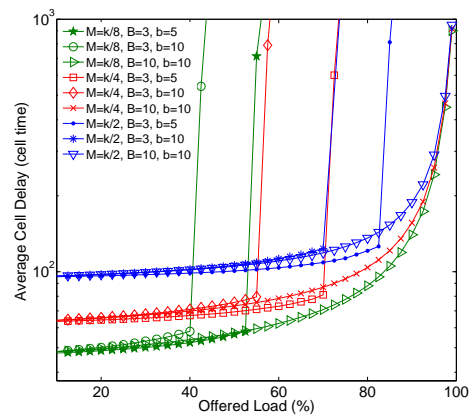


Figure 6.10: Delay performance of single-stage OQ-UDN 64×64 switch, under uniform Bursty traffic.

Uniform bursty traffic

In reality, workloads in the DCN are constantly changing. Distributed file systems in Big Data analytics, streaming media services and many other high-bandwidth demanding applications make the bursty traffic pattern prevalent

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

in a data centre network with high-levels of peak utilization. Therefore, it is useful to examine how a bursty traffic impacts the proposed switch performance. Figure 6.10 shows the latency of a 64×64 single-stage OQ-UDN under bursty traffic, for which the parameter M , the on-chip queues' capacity and the size of the burst are varied. Visibly, increasing M improves the switch throughput. Still, for $B = 3$ (minimum queues depth) and a burst size of 10 packets, the NoC structure saturates, and the blocking probability rises exponentially. Experimental results show that it is possible to ameliorate the switch response to burstiness by reducing the burst size. However, the throughput expansion is limited to about 14% (see Figure 6.10).

There is another way to boost the switch performance. Providing larger queues for the mini-routers proves much effective to resolve the saturation problem at the expense of some additional cost. On the whole, the standalone OQ-UDN as it is, do not scale with the switch size under bursty traffic. On the contrary, the multistage architecture shows robustness and flexibility.

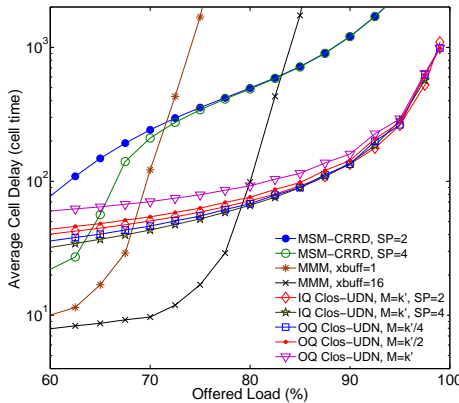


Figure 6.11: The average latency of MSM, MMM, IQ Clos-UDN and OQ Clos-UDN, for 256×256 switch size, under uniform Bursty traffic, $B = 3$.

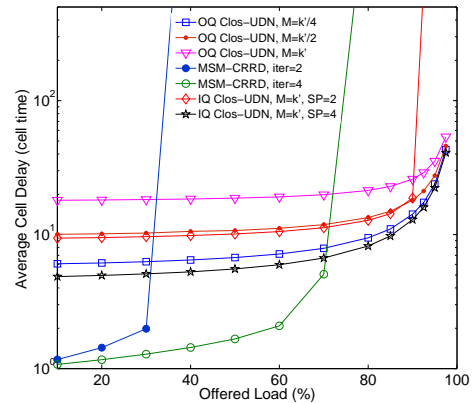


Figure 6.12: Delay performance of 64×64 MSM, MMM, IQ Clos-UDN and OQ Clos-UDN, under Unbalanced traffic, $B = 3$, $\omega = 0.5$.

Figure 6.11 illustrates the average end-to-end latency in MSM, MMM and IQ/OQ Clos-UDN switches. Under heavy loads, both semi-buffered and fully-buffered Clos architectures yield worse delay performance than the NoC-based switch with its two variants – IQ and OQ. The MMM switch cannot achieve full

throughput even when the internal buffers of the middle stage modules are worth of 16 packets, each. The flexibility of NoCs and the multistage interconnect used along with a dynamic RR packets dispatching work towards better distributing the traffic load and conserving high throughput. Next, the switch performance under non-uniform traffic arrivals is evaluated using computer simulations.

6.4.2 Unbalanced traffic

The performance of a 64×64 Clos with OQ-UDN modules is evaluated under unbalanced traffic whereby one fraction of the total input load is uniformly distributed among the switch outputs, and the other fraction goes to the output port with the same index as the issuing input port. If the unbalanced coefficient $\omega = 0$, then the traffic is perfectly uniform. On the other hand, if $\omega = 1$, then the switch deals with a directional traffic. Figure 6.12 shows the average packet delay for the different switching architectures with variable settings, input loads and values of $\omega = 0.5$. As for uniform traffic, OQ Clos-UDN switch outperforms the MSM switch with CRRD packet scheduling algorithm. Both IQ and OQ Clos-UDN switches can achieve comparable latencies if the parameters are adjusted (mainly speedup SP and M for the input-queued type, and M and B for an OQ-UDN module). Although both NoC-based designs are highly customizable, an input-queued structure with no speedup and full mesh depth ($M = k' = 8$) still do not respond well to traffic non-uniformity. Even when the on-chip interconnects run two times faster than the external line rate, the IQ Clos-UDN switch achieve offers a maximum throughput of 92%.

In Figure 6.13 ω is varied, and the behaviour of a 64×64 OQ Clos-UDN switch is observed as packet arrivals become more and more skewed. The simulation scenario comprises three traffic types: Uniform, hot-spot and diagonal. Note that the blocking probability in the central modules of the Clos switch evolves in the same way as the transferred packets ratio when the traffic shifts from a perfectly uniform to a skewed diagonal. When $\omega = 1$ (*i.e.*, diagonal traffic) and using the “*Modulo XY*” routing algorithm, packet flows travel horizontally in the NoC modules towards $LC(r, j)$ links. Averaging the proportion of packets over all nodes, results in a fraction being equally distributed among the rows

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

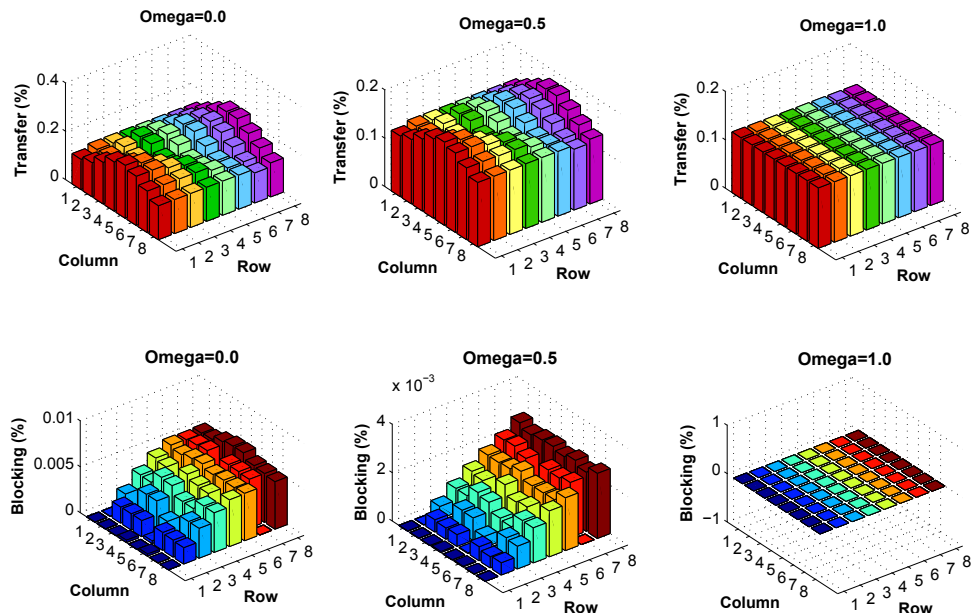


Figure 6.13: Variation of the transfer and blocking probability in the central modules of a 64×64 switch.

of the OQ-UDN module. For a hot-spot traffic where $\omega = 0.5$, packets cross northern and southern links to reach the outputs of the UDNs. This explains the disproportion of the load sent in the X and Y directions. The amount of traffic crossing the NoC-based modules doubles under uniform traffic, and that the blocking probability gets less equalized across the mini-routers.

This thesis presents a class of multistage switching architectures that have been designed to respond to stringent requirements of large-scale DC networks. The next section is concerned with scalability in connection with high-performance. The two aspects can be mostly inferred from the switch throughput and delay variations as the switch parameters and the simulation scenarios are altered.

6.4.3 Scalability of the switch

The throughput stability is fundamental. It provides clear evidence of the resiliency of the switching architecture as the traffic fluctuates and becomes

6.4 Performance evaluation

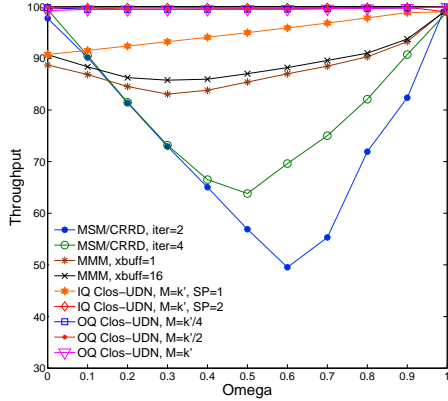


Figure 6.14: Throughput stability of 64-ports switches.

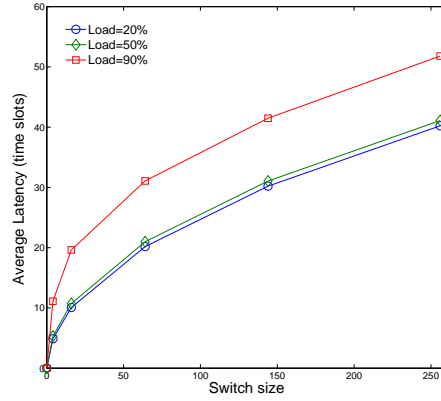


Figure 6.15: Impact of the switch size on the packet delay under hot-spot traffic.

more harsh. With the help of small on-chip queues in the CMs of the Clos switch, the incoming traffic can be absorbed and transferred from one stage of the NoC to the subsequent stage. Figure 6.14 depicts that the IQ Clos-UDN with full depth ($M = k' = 8$) and $SP = 1$ achieves 90% throughput. A buffered MMM architecture provides better throughput than MSM with CRRD scheduling (60% throughput if $iter = 4$ and $\omega = 0.5$). In the mean time, the OQ Clos-UDN switch offers full throughput under the whole range of ω even for minimum-value settings ($B = 3$ and $M = k'/4 = 2$ for a 64-ports switch). DCN switches/routers must fulfil the large-scale in addition to the data-intensive communication prerequisites. In this context, the impact of the switch size on the end-to-end packet latency is inspected. The switch valency is varied from 4 to 256, and the overall packet delay is measured in the OQ Clos-UDN switch under light loads (20%), medium loads (50%) and heavy loads (90%). Figure 6.15 shows the following: The latency variation under light and medium traffic loads is approximately the same, regardless of the switch port count. Observe that when the switch is heavily loaded, the latency increases slowly with the switch size. Yet, it does not exceed 50 time slots when the load is as high as 90 percent. In the following part, the analytical results are compared to outputs of the simulation.

6.4.4 Accuracy of the analytical model

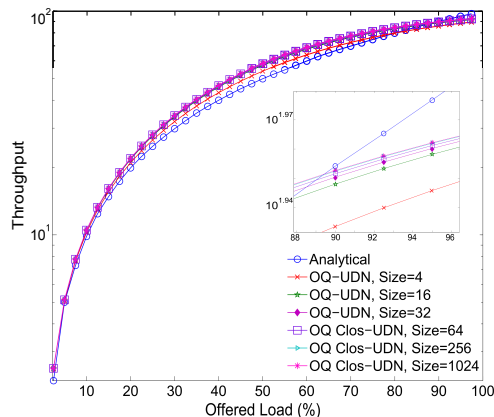


Figure 6.16: Average throughput for various switch valency, under Bernoulli uniform traffic.

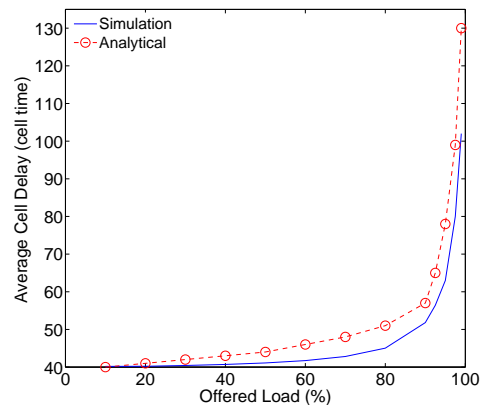


Figure 6.17: The average end-to-end delay in a 256×256 switch under Bernoulli uniform traffic.

Next, the analytical and simulation results are compared for different switch sizes while we set the output buffers capacity B to 3 packets, each. Figure 6.16 shows the variation in the throughput percentage under Bernoulli uniform arrivals. The proportion of throughput increases linearly when the input load increases because of the number of packets generated in the system. The values obtained using analytical model approach those of simulation and that under light loads, simulations perfectly match the analytical model. For a single-stage OQ-UDN switch, the deviation is about 4.5% as moderately medium traffic loads come to the switch inputs. This difference margin increases with the number of ports getting higher and with the traffic load becoming heavier. According to Figure 6.16, the larger is the switch size, the bounded becomes the approximation. The fact that the analytical model drops some architectural considerations, partially accounts for this lack of accuracy. However, the disparity between the analytical and the simulation results still do not go beyond 7.98% for the smallest single-stage switch of size 4-ports and 5.2% for a 64-ports multistage OQ Clos-UDN.

Using the value of the input load, the average arrival rate to the OQ Clos-UDN switch inputs λ can be calculated. In the steady state where all

inputs perpetually have pending packets, the service rate at the input FIFOs is restricted to the availability of the first output buffer that they request in the selected CM. This probability is given by $P_{fwd} = 1 - P_{ctr}$. It can be dynamically calculated whenever information about the arrival and departure probabilities in a single on-chip node is provided. At the steady state operation mode, it is easy to calculate P_{dep} . Every output queue serves one of the n associated inputs with equal probability making $P_{dep} = 1/n$. Figure 6.17 shows the delay variation for a 256-ports switch for full mesh depth, $B = 3$ and uniform packet arrivals. It demonstrates that the simulation results decently support the analytical model.

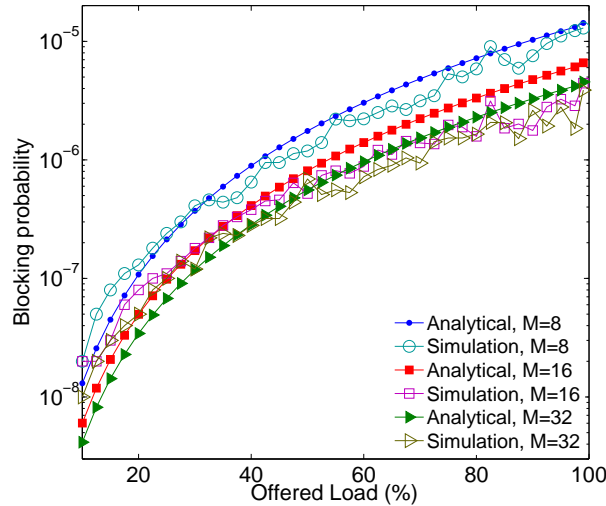


Figure 6.18: Variation of the blocking probability in 64×64 switch under Bernoulli uniform traffic.

The last set of simulations is performed to trace the end-to-end blocking probability in the OQ Clos-UDN switch. Given the multistage switching topology and the assumptions that are made for the analytical analysis, the central modules show to be the one and only bottleneck of the design. Hence tracing $B(r)$ in the central stage modules, indicates the same blocking attitude as for the OQ Clos-UDN switch. In the following experiment, a fabric of size 64×64 is considered. It can work in a single stage or be plugged into the middle stage of the Clos architecture for larger switch valency. The parameter

6. A MULTISTAGE SWITCHING FABRIC WITH OUTPUT-QUEUED NOC MODULES

B is set to 3, and *Bernoulli* uniform packet arrivals are considered. Note that the blocking probability rises exponentially with the input load, and that the proposed mathematical approximation of $B(r)$ approaches the simulation curves as depicted in Figure 6.18. For light workload intensities, the network is almost not blocking. When the switch is lightly loaded (load < 30%), the simulation results are located above the analytical model. This is mainly because the OQ Clos-UDN switch is not overloaded and that packets travel flawlessly across the NoC fabric without a noticeable blocking. Therefore, the model do not correlate with the simulations. As the traffic loads become heavier, the blocking likelihood of the switch rises and the analytical starts reflecting the real behaviour. Irrespectively of the number of pipeline stages, M the blocking probability $B(r)$ remains less than 10^{-5} .

6.5 Summary list

- 1- A multistage packet-switching architecture with an output-queued NoC fabric is proposed.
- 2- Unlike the IQ Clos-UDN, the OQ type switch needs no speedup to offer high performance.
- 3- The switch achieves high throughput by means of small capacity on-chip routers which can be readily implemented using the current technology.
- 4- Using small on-chip buffers and a buffered flow-control mechanism, the limited buffering space in the OQ NoC fabric is managed and overflows are prevented.
- 5- Using extensive simulations, the multistage OQ Clos-UDN switch shows to scale well with the switch valency and the traffic load and type.
- 6- An analytical model for the performance metrics of the switch is provided. The switch throughput, the end-to-end packet latency are modelled, and an estimation on the upper bound of the blocking probability in the OQ NoC fabric is given.

6.6 Conclusions

In a first part of this chapter, the OQ Clos-UDN switching architecture is described. It is a highly-scalable multistage switch with OQ NoC fabric that provides high-performance, scalability and parametrization. Besides, the switch performance is assessed using simulations. It is shown that the OQ Clos-UDN achieves high throughput under a variety of traffic types. An analytical approximation is provided for the theoretical throughput using the queueing theory and Markov chains. Experimental results revealed that the average deviation of the model is about 7.9% for a single stage OQ-UDN (size < 64), and is around 5.2% for a Clos-UDN switch (size ≥ 64). Also, an approximation for the overall packet delay is provided, and an upper bound on the end-to-end blocking probability in the central modules of the OQ Clos-UDN switch is given.

7

Clos-MDN: A multistage packet-switch with multi-directional NoC fabric

7.1 Introduction

Motivated by the shortcomings of the previous works, a three-stage Clos-network switch with an MDN fabric [154] is proposed. The switch is called Clos-MDN. The first contribution takes place at the heart of the Clos-network where the conventional crossbars are replaced with MDN modules. An MDN is an optimized version of the UDN switch where the space design is better explored for less cost implications. A single MDN module is a regular 2D mesh NoC where I/O ports are equally distributed among the four sides of the peripheral. It implements VCs and a buffered flow-control to assure *East/West* and *West/East* traffic flows with no deadlocks. Also, the Clos-MDN switch is fitted with bidirectional cross-interconnections linking the middle-stage's elements. This significantly extends the switching facility between the CMs, and makes the architecture a wrapped-around three-stage Clos-network. The second main contribution is about implementing a congestion-aware routing algorithm to adaptively distribute the traffic load among the CMs. Consequently, the Clos-MDN allows to deal better with skewed traffic, and to intuitively uplift the overall DC network performance.



The Multi-Directional NoC switch

The single-stage MDN switch was introduced in [151] as an extension of the UDN proposal [154]. They both have common features, but the MDN design tends to efficiently make use of the NoC concept in building a compact packet-switch. An MDN is a regular 2D mesh of generic size $N \times N$. The set of input/output pads are placed on the perimeter of the NoC layout as shown in Figure 7.1. The MDN can be thought of as the concatenation of two UDN switches where packets can flow horizontally in two opposite directions. The MDN switch implements a buffered credit based flow-control, and adopts the store-and-forward switching mode.

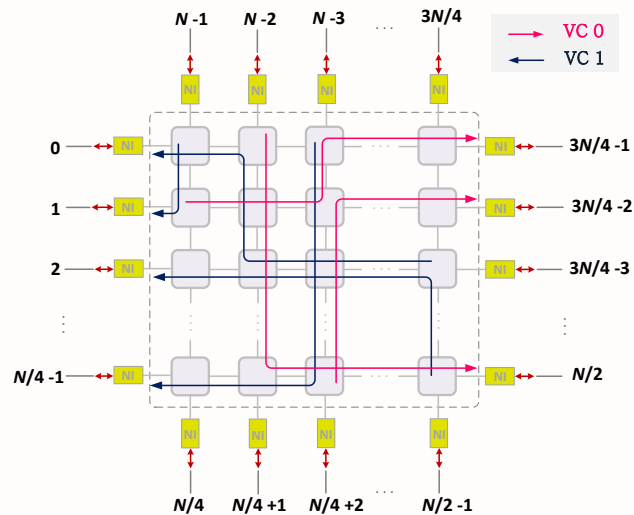


Figure 7.1: A high-level diagram of the central-stage MDN module.

To avoid deadlocks, two VCs are implemented to separately convey the *East/West* and *West/East* traffic. Packets cross the first virtual channel VC0 if their corresponding output destination is located eastern to its input port. The second channel VC1 is used if the packet destination is located western to the input port. Input queued mini-routers are equipped with small crossbars and RR arbiters to resolve the input contention. Figure 7.5 depicts a high-level diagrams of the different mini-routers used in the MDN fabric.

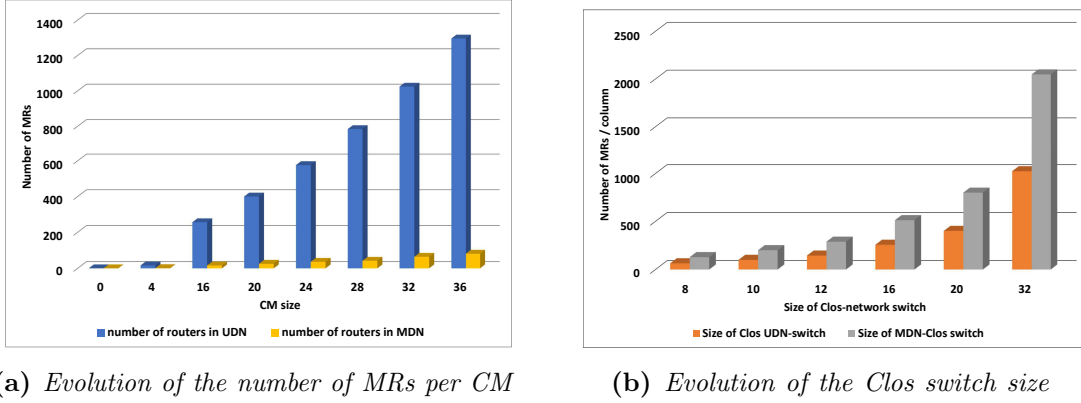
7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

The Clos-MDN switch has several architectural and scheduling advantages over the conventional MSM and MMM switches as well as the Clos-UDN architecture.

- It Obviates the need for complex and costly input modules, by means of few, yet simple, input FIFO queues
- It avoids the need for a complex and synchronized scheduling process over a high number of input/output modules and port pairs
- It provides speedup, load balancing and path-diversity thanks to the NoC-based fabric nature
- It allows the switch size to grow faster than with UDN modules for less design cost
- It deals better with skewed traffic thanks to the inter-CM links and the adaptive routing scheme

7.2 Motivation

The UDN and MDN modules have similar architectures. They both have input-queued mini-routers on the mesh grids and they both adopt the store-and-forward switching mode and a buffered flow-control mechanism to monitor the packets forwarding across the NoC fabric. A round robin scheme is used to resolve the input contention among packets. Eventually, the UDN and MDN switching modules can be designed in the same way. The network interface, flow-control as well as the input buffers of the mini-routers are the same. Still, on-grid routers of the two switches differ in number of ports which implies changes in the high-level design of the MDN switch in comparison with UDN. The first apparent difference between a UDN and an MDN design reports to how the set of input/output pins is placed in the block layout. In a UDN module, input pins are located at the west side of the layout whereas all output pins are placed at the east side. As for the MDN switch, input and output pins are set one next to the other. They set of I/Os is equally distributed among the four sides



(a) Evolution of the number of MRs per CM

(b) Evolution of the Clos switch size

Figure 7.2: Requirements of the Clos-UDN/Clos-MDN switches.

of the peripheral. In previous work [153], single-stage UDN and MDN switches have been implemented on the specific FPGA device *Virtex 5 – XC5VTX240T, FF1759, – 2*. The synthesis has been done in an ASIC 65 nm CMOS technology whereby results showed that the critical path is on the mini-routers with the highest degree (degree 3 for an UDN and degree 4 for an MDN module). It starts from the RR registers of any arbiter, passes through the flow-control logic and then the *Read Enable* signal that determines the occupancy status of the buffer. Based on the geometry of every NoC module, the requirements of the Clos-UDN and Clos-MDN switching architectures are computed, respectively. In the previous chapters, it was proven that the Clos-UDN switch variations can offer high performance when fairly sized NoC modules are implemented. In the following, the total number of embedded mini-routers in regular¹ UDN and MDN fabrics is estimated. As shown in Figure 7.2(a), the number of on-chip nodes grows fast in a UDN module. It goes beyond a practical limit even when the switch size is moderate. For instance, a CM with 32 I/Os that can make part of a 1024×1024 Clos-UDN switch, needs 1024 mini-routers per central module. This is to be compared with only 64 nodes figuring in an MDN module of the same size Clos-MDN architecture. Figure 7.2(b) shows the evolution of

¹A regular 2D mesh where the number of pipeline stages – M – is equal to the number of I/O ports of the module.

the Clos-network switch size as function of the MRs per one column in the mesh layout of the UDN/MDN modules. Clearly, both figures affirm that an MDN-based design is more compact.

Motivated by the features of the MDN design, a multistage packet-switching architecture that scales fast in size and load fluctuation is proposed. In the rest of the chapter, the terminology of the Clos-MDN switch is described, and its performance is tested under a range of traffic patterns.

7.3 High-level architecture

In this section the switching architecture is presented. First, the topology and the packet buffers distribution in the switch are outlined. Later on, a detailed description for the MDN modules is provided. Also, the packet dispatching process and the routing algorithm inside and in-between the MDN central modules are given.

7.3.1 Network topology and packet buffers

The middle stage of the Clos-network is altered. Instead of the common point-to-point connection crossbars, multi-directional NoC modules are plugged and the packet buffers organization in the IMs and OMs is updated as follows: The first and second stages of the switch architecture are made of k Input/Output Modules (IOMs), each of which is of size $n \times n$. The input and output ports of the Clos-MDN switch are spread on the edge modules in opposite directions.

Figure 7.3 shows the generic layout of the Clos-MDN switching architecture, while Figure¹ 7.4 depicts an example of an 8×8 switch. The green queues on Figure 7.3 are for the input buffers and blue queues show the output port buffers of the Clos-MDN switch. Every IOM regroups n input FIFOs, each of which is associated to one input port. It can receive at most one packet and sends at most one packet to a central module at every time-slot. There are also n output queues per IOM each is associated to one output port. It can

¹To have a quick understanding on how the input and output port buffers are distributed in the Clos-MDN architecture

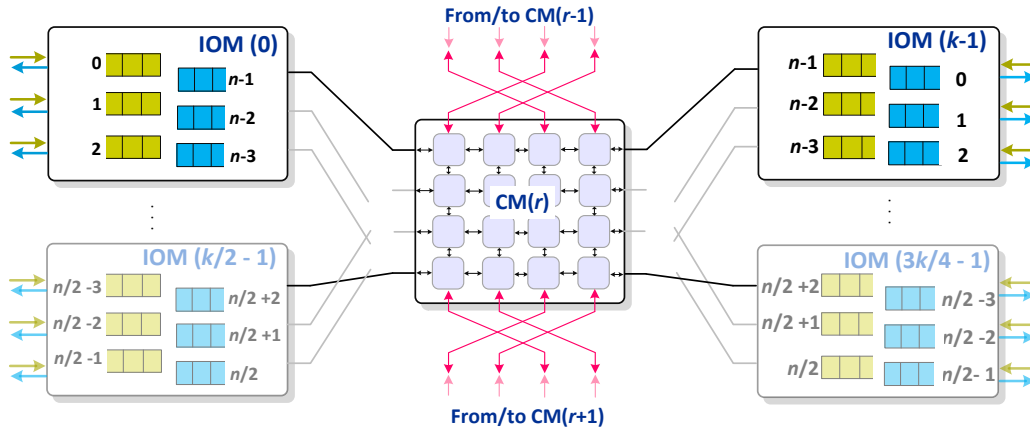


Figure 7.3: Generic layout of the Clos-MDN switching architecture.

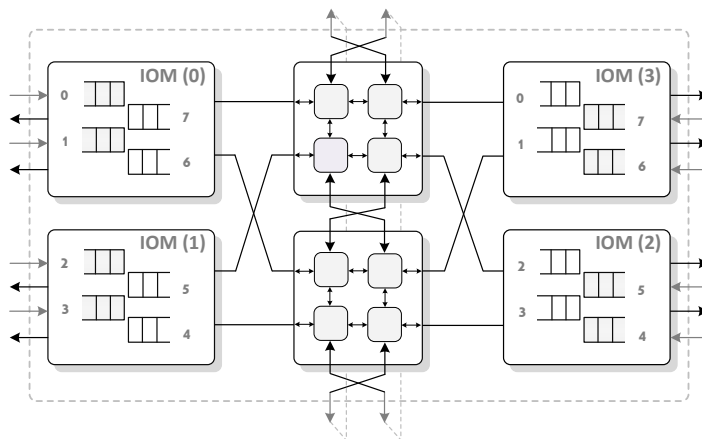


Figure 7.4: An example of an 8×8 Clos-MDN switch.

receive at most m packets (from the different MDN blocks), and forwards one packet to the output line card at every time-slot. The middle stage is made of m MDNs, each of dimension $k \times k$. For an arbitrary non-blocking Clos-network, the number of outlets in any of the first-stage modules (m) can differ from the number of its inlets (n). However, the simple case *Bene's* network for which $n = m$ is considered. This makes the Clos-MDN switch architecture, the lowest-cost rearrangeably non-blocking Clos-network, and avoids the need for a sophisticated insertion policy to distribute packets among input buffers at the

7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

traffic arrival phase.

The MDN implements two virtual channels to avoid deadlocks. Two configurations are considered in testing the performance of the Clos-MDN switch¹. At a first time, the performance of the switch is tested for an asymmetrical buffer distribution among virtual channels, whereby west mini-routers have 2/3 of the buffer depth for VC0 and 1/3 for VC1 and east mini-routers use 1/3 of the port buffering space for VC0 and 2/3 of it for VC1. Besides, a symmetrical buffer distribution among the two VCs is considered. Figure 7.5 shows a high-level diagram of the mini-routers depending on their location on the grid and the buffering space organisation.

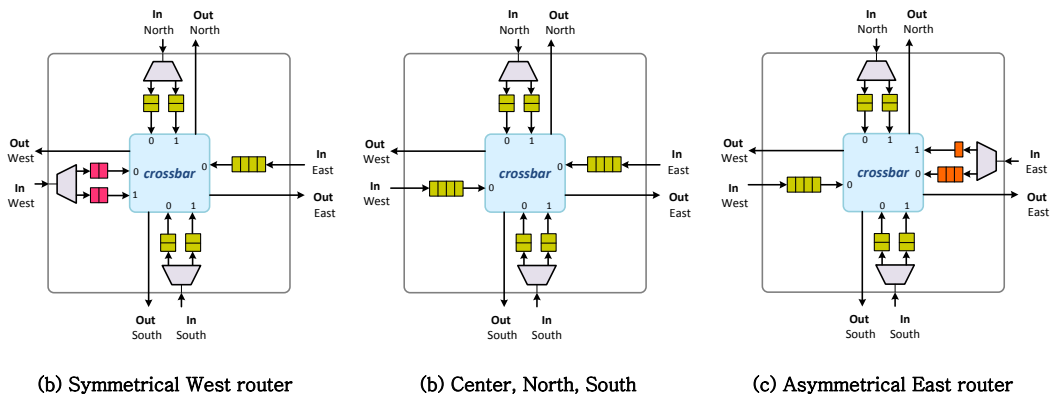


Figure 7.5: Symmetrical/asymmetrical buffers distribution in the MDN mini-routers.

7.3.2 Packet routing in the MDNs

A static packet dispatching scheme from the IOMs is considered, for which every input FIFO constantly delivers packets to the same MDN module on the connecting link. Traffic flows do travel in all directions in the central stage modules until the external links bridging the corresponding IO modules. Based on their ultimate output ports, packets are transferred from a pipeline stage to the next one inside the MDN modules. In the rest of the chapter, the expression *local output port* refers to the outlet within a CM module, and the term *ultimate*

¹Refer to Section 7.4.

output port is reserved for the Clos-network egress port, through with the packet exits the switch.

Packets are routed within a central MDN as following: The first step consists on finding out the IO module index to which is related the packets' ultimate output port. Upon their entry to a CM, packets are locally routed using a combination of two algorithms: “XY” [157] algorithm and the “MDN Modulo” routing [154]. The “XY” algorithm has been long ago introduced for mesh NoCs. As for the MDN fabric, it is used to route packets in the mesh network whenever the local output port is perpendicular to its input port. It simply starts by forwarding packets horizontally to the correct column (x-coordinate) and then vertically to the right row (y-coordinate). The “MDN Modulo” algorithm is an improved version of the basic “XY”. It introduces an extra turn in one intermediate column before the last one, to better balance the traffic across the mesh. It is used in the MDN switch if the local input and output ports are parallel.

7.3.3 Proactive congestion control

The previous results in Chapter 4, showed that a static packet dispatching and an oblivious routing scheme are irrelevant to skewed traffic arrivals [193, 194]. In fact, the NoC-based switches can get congested under some traffic patterns causing the packet delay to become longer, and the switch throughput to deplete. Therefore, the central-stage modules of the Clos-MDN switch are made capable of sharing the traffic load via intermediate links. Also, two virtual channels are used on each link to transport packets depending on the flow direction. This conserves the packets' flowing direction in any CM, and prevents deadlocks.

The additional connections extend the advantage of the NoC geometry to the Clos-network, and make the multistage switch architecture a wrapped-around network. The $CM(r)$ is connected to $CM((r - 1) \bmod m)$ and the module $CM((r + 1) \bmod m)$ by means of $\frac{N}{4}$ interleaved links as depicted in Figure 7.3 and explained in the Algorithm 3. Note that the term $MR^r(a, b)$ refers to the mini-router in module $CM(r)$ located in row a and column b of the 2D mesh. Choosing an interleaved configuration is made to ensure that sending

7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

Algorithm 3: Inter-CM interleaved connections

- 1: **for** $r \in \{1, \dots, m-1\}$ **do**
 - 2: $r' \leftarrow ((r+1) \bmod m)$ and $r'' \leftarrow ((r-1) \bmod m)$
 - 3: **for** $i \in \{1, \dots, k-1\}$ **do**
 - 4: $j \leftarrow ((\frac{k}{2} + i) \bmod k)$
 - 5: $\text{MR}^r(k-1, i)$ connects to $\text{MR}^{r'}(0, j)$
 - 6: $\text{MR}^r(0, i)$ connects to $\text{MR}^{r''}(k-1, j)$
-

packets away from their original congested CMs to neighbouring modules does not increase the remaining hops count¹.

The process of packet routing throughout the CM modules is subject to some constraints. The ultimate goal is to maximize the switch throughput under coarse traffic without affecting the delay performance. Therefore, a suitable metric is used for the routing scheme to match with the global Clos-network congestion status while being inexpensive to compute. The RCA concept [173] is adopted to evaluate, and to propagate the congestion information across the central module of index r and its direct neighbours (blocks of indexes $((r-1) \bmod m)$ and $((r+1) \bmod m)$). The congestion metric weights both distance (hops count until the exit port) and buffers occupancy to make sure that the traffic is adaptively transferred through minimal paths and that the average packets delay is little affected by the inter-module routing decision. As it was defined in Chapter 5, a routing quadrant is the sub-network limited by the packet's current position in the MDN mesh and the egress port through which it exits the current CM to its corresponding IOM. Note that the local CM information is the one readily available at a given CM module and representing the congestion status of the on-chip nodes in a given routing quadrant. Given its current position, a packet can travel in one of four quadrants² N/E , S/E , N/W and S/W with each quadrant having exactly two possible output directions excluding the current local port. As with the previous proposed packet-switching

¹In the worst case scenario, a packet will do the same number of hops in the neighbour CM as it would have in its non-congested CM for two reasons: First, the inter-module routing algorithm considers the distance metric and second packets are minimally routed within a single MDN.

² N : North, S : South, E : East, and W : West.

architectures, the performance of the Clos-MDN switch is assessed under different design settings and traffic scenarios.

7.4 Performance Analysis

In this section computer simulations are used to evaluate the performance of the Clos-MDN switch and to compare it to state-of-the-art switching architectures. As in previous chapters, the simulation models are built on top of an event-driven simulator written in C language. For the following experiments, a wide range of workloads is considered

1. Bernoulli/bursty uniform
2. Bernoulli/bursty hot-spot arrivals
3. Diagonal traffic

Note that for all simulations, The switch size is 256×256 and the capacity of the on-chip input buffers (buff) in the Clos-MDN switch is 4 packets each, unless it is otherwise stated. The first set of simulations is performed under uniform traffic. Packets are assumed to have the same fixed size. For the sake of comparison, the input buffers capacity of the embedded mini-routers is made the same for the Clos-UDN and Clos-MDN switches. The Clos parameters n, m are made the same for both switches configurations, in which case the performance disparity is mainly attributed to the NoC modules. The essence of the Clos-MDN is in the prospect of building high-capacity switching architectures with small sized NoC modules. Note that for any switch valency, a central-stage UDN¹ block uses sixteen times as many mini-routers as an MDN module employs.

7.4.1 Uniform traffic

First, the performance of the Clos-UDN, the congestion-aware Clos-UDN, and the Clos-MDN proposals are compared. Figure 7.6(a), shows that the Clos-UDN

¹For full mesh design where the number of the unidirectional NoC stages is equal to the number of inlets/outlets [176].

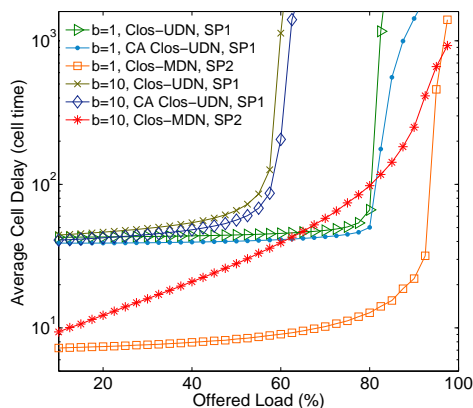
7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

switches have much higher packet latency than the Clos-MDN switch under both uniform *Bernoulli* and uniform bursty arrivals (curves with burst size b set to 1 and 10 respectively). Remember that the initial delay ties in with the number of NoC stages that packets need cross until exiting the central modules. After a number of time slots, the pipeline is filled in, and the packet latency variation with the traffic load becomes quasi constant. Note that with less on-chip mini-routers and a $SP = 2$, the Clos-MDN switch outperforms a Clos-UDN switch that only relies on larger NoCs (*i.e.*, full mesh UDNs and $SP = 1$). This attests of the efficiency of the Clos-MDN design in terms of area, especially that it is not expensive to run short on-chip links a bit faster. The bursty uniform traffic can be modelled as an ON/OFF process with a geometric distribution and a given burst size b . A burst of b packets that come to the same input port of the switch during the ON period are destined to the same output port. Figure 7.6(a) depicts the simulation results for $b = 10$. Observe that the NoC-based switches perform under uniform bursty traffic in a similar way as they behave under Bernoulli arrivals. Rising the speedup – SP – improves the Clos-MDN switch’s performance. Even so, the switch still cannot achieve full throughput (82%). Overall, trading the area by speedup ameliorates the Clos-MDN switch performance. Compared to the Clos-UDN switch (and the CA Clos-UDN switch, respectively) Clos-MDN shows an improved throughput – approximately 20% (10%) under Bernoulli traffic and 30% (27%) under uniform bursty traffic arrivals. Next, the switches performance is evaluated under non-uniform packet arrivals.

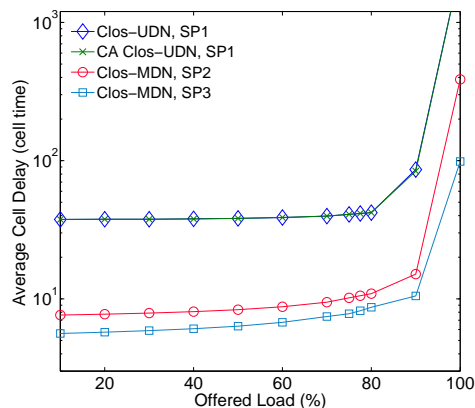
7.4.2 Non-uniform traffic

Non-uniform traffic is described by an unbalance degree $\omega \in [0, 1]$. The parameter $\rho_{i,j}$ denotes the normalized load from input i to output port j . It is given by $\omega + \frac{1-\omega}{n-k}$ when $i = j$ and $\frac{1-\omega}{n-k}$ otherwise. The traffic is uniform when $\omega = 0$ and directional if $\omega = 1$ (packets are always destined to only one output port). Any intermediate value of ω implies that the traffic is a weighted mix of uniform and directional traffic also called unbalanced traffic. The next simulation set is performed to test the Clos-MDN switch tolerance to hot-spot traffic ($\omega = 0.5$). In Figure 7.6(b), the average packet delay of the Clos-UDN/MDN switches

7.4 Performance Analysis

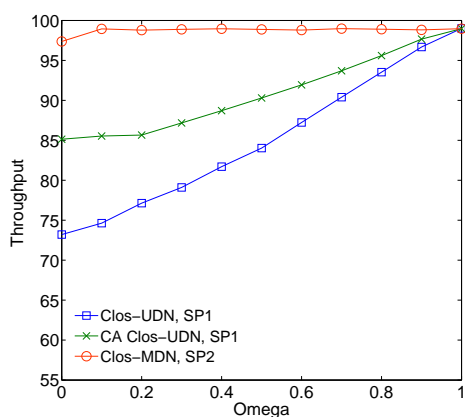


(a) *Uniform traffic*

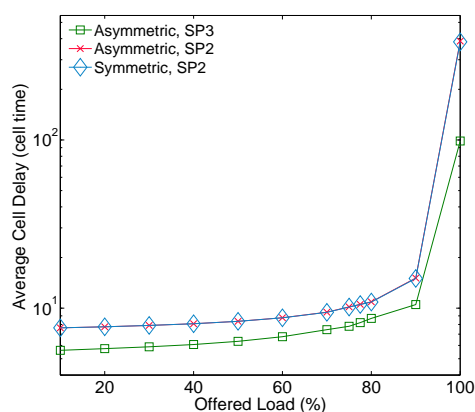


(b) *Bernoulli hot-spot traffic, $\omega = 0.5$*

Figure 7.6: Performance for 256-ports Clos-UDN/MDN Switches.



(a) *Throughput stability*



(b) *Delay variation of the Clos-MDN*

Figure 7.7: Performance for 256-ports switches under Bernoulli hot-spot traffic, $\omega = 0.5$.

is plotted for different speedup factors. Switches with UDN modules perform better than the Clos-MDN if the internal NoC connections run as fast as the external line rate. However, a speedup of two suffices to noticeably reduce the packet delay of the Clos-MDN switch, and to make it achieve full throughput as clear in Figure 7.7(a). Bursty traffic is relevant to DCNs [171]. In the following, a further study is conducted to test the scalability and the robustness of the MDN-based multistage design under this traffic pattern, by varying its architectural settings. The effect of speedup, load, on-chip buffers capacity,

7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

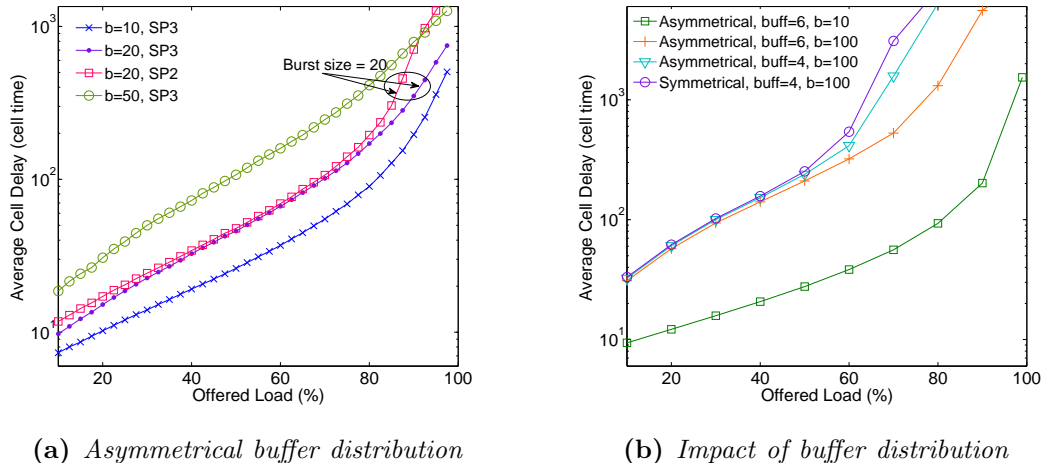


Figure 7.8: Delay performance for 256-ports Clos-MDN Switches under non-uniform bursty traffic.

buffers distribution, and the switch valency are investigated. Increasing the burst size entails more packet delay and throughput degradation, as shown in Figure 7.8(a) and Figure 7.8(b). Two conclusions can be drawn: First, increasing the speedup factor boosts the switch performance, even though large bursts of packets break in the switch. Second, increasing the on-chip buffers capacity (from 4 packets to 6 packets, each, in the simulations), also lifts up the Clos-MDN performance. Under bursty non-uniform traffic, the Clos-MDN performs a little bit better under unbalanced traffic, when the buffering capacity is asymmetrically distributed among VCs. However, this proves to have no remarkable effect under hot-spot traffic as Figure 7.7(b) shows.

Now, the delay and throughput performance of the Clos-MDN switching architecture are compared to those of a bufferless and a buffered Clos-network switches (the MSM with the CRRD algorithm [64] and the MMM switch as described in [75]). Figure 7.9 depicts the simulation results for the three switching architectures with the minimum optimal settings¹. Obviously, the current proposal fits in the buffered Clos architectures category. But comparing

¹The MSM switch is tested with 2-iterations CRRD matching since even with larger iterations the switch performance converges to nearly the same values [64]. The MMM crosspoint buffers are set to 16 packets each, as with only one-packet crosspoint buffering the switch throughput do not exceed 65% under bursty traffic [75].

7.4 Performance Analysis

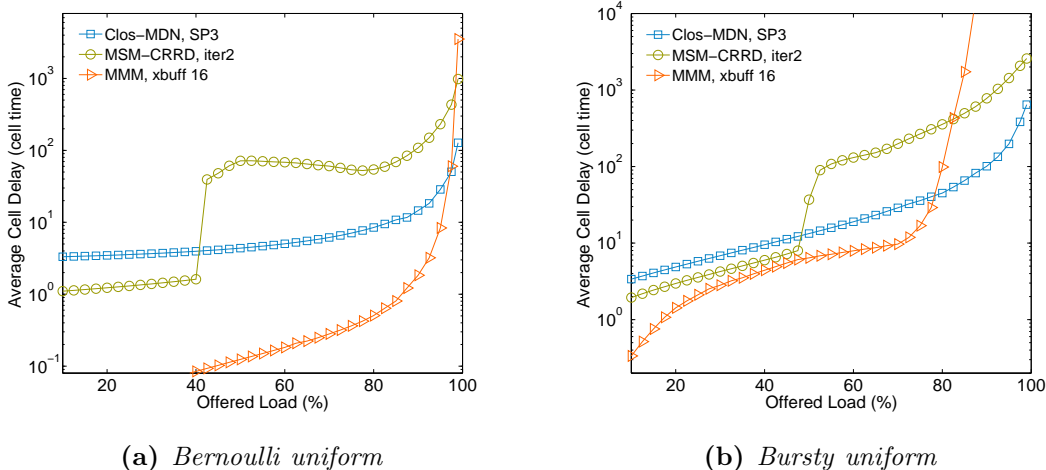


Figure 7.9: Performance of 256-ports MSM, MMM and Clos-MDN Switches.

its performance to the baseline bufferless MSM switch, helps situate the Clos-MDN and analyse its response to the traffic arrivals with respect to its features (number of packet buffers and their capacity, scheduling complexity, etc.). In Figure 7.9(a) is shown the average packet latency for switches of size 256×256 . The following conclusions can be drawn: A bufferless switching architecture performs well under light to medium loads however the delay rises sharply at around 40% load, and it never pulls down. The MMM switch also provides low latency and outperforms the MSM and Clos-MDN switches under light to medium loads. This is attribute of its large capacity crosspoint buffers and – also – due to the absence of many stages (one stage to transfer packets to their related OMs). The Clos-MDN switch experiences a relatively higher latency than other switches under light to medium traffic loads. As it was explained before, the pipelined structure of the NoC-based central modules is on behalf of this initial cumulative delay. However, it is interesting that the delay variation of the Clos-MDN switch is quasi stable showing a good scalability to the load fluctuation. In Figure 7.9(b), uniform, identical and independent reference pattern of packet bursts arrivals is considered at the different multistage switches inlets. Both MSM and MMM switches yield better latency than the Clos-MDN switch under light loads. However the MSM reiterating the CRRD matching two times, experiences an abrupt delay increase (at 55% load). Besides, the performance

7. CLOS-MDN: A MULTISTAGE PACKET-SWITCH WITH MULTI-DIRECTIONAL NOC FABRIC

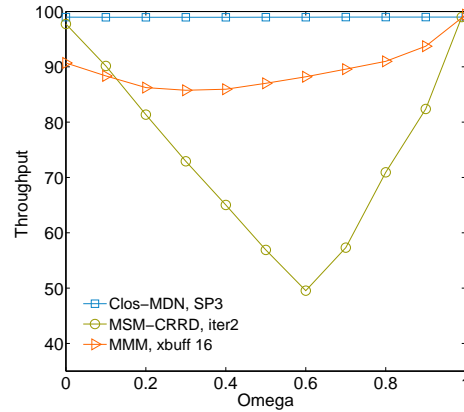


Figure 7.10: Throughput performance of 256-ports MSM, MMM and Clos-MDN Switches.

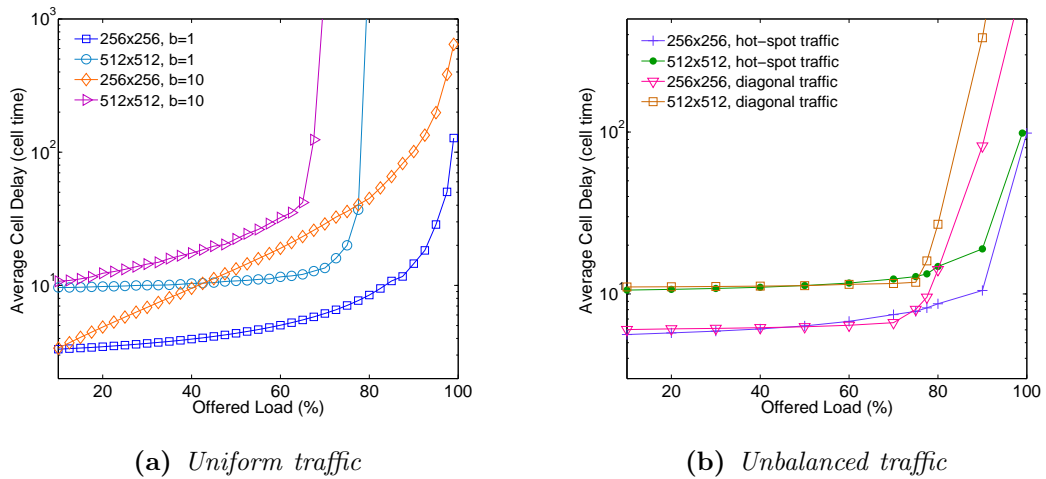


Figure 7.11: Impact of switch size on performance of Clos-MDN Switch, $SP = 3$.

of the MMM switch degrades under bursty traffic, whereas the delay variation of the Clos-MDN is near constant. In Figure 7.10 the throughput variation of the different switches is plotted as the unbalance degree of traffic, ω is varied. Note that the MMM switch experiences less performance fluctuation than the MSM for which the throughput drops drastically (50% - 55% for $\omega \in [0.5, 0.7]$). Setting the speedup factor to 3, makes the throughput of Clos-MDN full across the entire spectrum of ω .

Next, the performance of the Clos-MDN switching architecture is tested by varying the port count, on-chip buffers distribution and capacity, and the

traffic type. Figure 7.11(a) depicts plots for the end-to-end packet latency for respectively 256 and 512-ports switch under uniform arrivals and a speedup factor $SP = 3$. A speedup of three proves enough for a 256×256 Clos-MDN switch to achieve full throughput. Yet, it is still insufficient to ameliorate the performance of a 512-ports switch. Setting up the NoC speedup boosts the switch performance. However, it does not resolve the persistent backlogs that can form inside the MDN modules under heavy traffic loads. Figure 7.11(b) shows the delay curves for two non-uniform traffic patterns: Hot-spot and diagonal. The latter is a very skewed traffic that is more difficult to schedule than any uniform loading. With a $SP = 3$, the Clos-MDN switch still performs well under the diagonal traffic, and achieves full throughput under hot-spot loads.

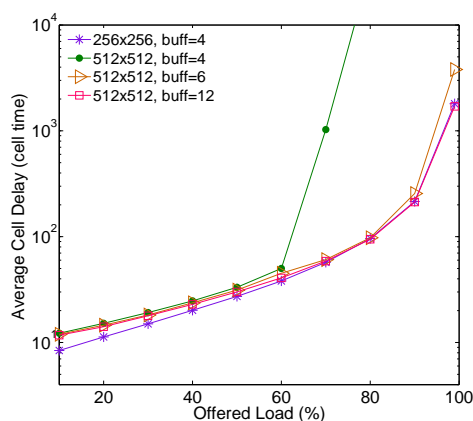


Figure 7.12: *Impact of switch size on performance of Clos-MDN Switch, $SP = 3$, Bursty non-uniform.*

The last set of simulations is performed under bursty hot-spot arrivals. Figure 7.12 shows that increasing the switch valency deteriorates its response. Using a $SP = 3$, the throughput of a 512-ports switch is still bounded to 76% whenever the on-chip input buffers capacity – buff is 4 packets. Actually, skewed and heavy loads of packets arriving in bursts to the switch inlets produce backlogs, and translate into throughput deterioration. Other than speeding up the NoC fabric, extending the input buffering space of all mini-routers to 6 packets, each, highly enhances the throughput performance. However computer

simulations show that there is little interest in further increasing the buffers capacity (the switch throughput converges with $\text{buff} = 6$ and there is little delay improvement if the parameter buff is set to 12 packets).

7.5 Summary list

- 1- The Clos-UDN switch varieties exploit unidirectional NoC modules to build a scalable fabric. The congestion-aware Clos-UDN switch tolerates inclined traffic – better than the primitive version of the Clos-UDN switch described in Chapter 4.
- 2- Yet the central-stage switching elements allow one horizontal direction to forward traffic from the input line cards to the output line cards.
- 3- The experimental results have shown that the UDN modules require a minimum expansion factor to offer high performance. This translates into more pipeline stages, and makes the design area of the central NoC fabric grow linearly with the multistage switch size.
- 4- The NoC concept is further exploited and a multi-directional NoC fabric that improves the design efficiency, cost and performance is used.
- 5- The MDN fabric has the potential to furnish a large valency multistage switching architecture. By means of compact CMs, it is possible to augment the switch port count while small-sized MDNs are put to use.
- 6- Virtual channels are implemented to prevent the deadlocks. The traffic is distributed among the channels based on the source/destination pair of addresses of each packet.
- 7- The traffic travels in all directions across the Clos-MDN switch. Packets can fly one congested CM to a less encumbered neighbour via the interleaved CM connections.

- 8- The traffic load is proactively spread over the Clos-network modules with reference to their congestion status. This approach is known as micro load-balancing, and is convenient to amend the switch performance under critical arrivals.

7.6 Conclusions

In this chapter, a three-stage Clos-network switch with multi-directional NoC modules – MDNs – is designed to overcome some of the shortcomings of the conventional crossbar-based multistage switches. Adopting MDNs obviates the need for complex and costly buffering structures at the input modules of the switch. It also avoids complex and synchronized scheduling processes that the bufferless Clos switches need and large crosspoint buffers which common MMM switches require. Compared to the Clos-UDN switch, the Clos-MDN switch scales better in size and load fluctuation. Thanks to the efficiently designed MDN modules, the architecture offers a range of settings that can be tuned to sleekly achieve given performance involving the minimum possible cost and complexity. In Appendix C, an initial work on the analytical model of the Clos-MDN switch performance is provided.

8

Conclusions

In this research, contributions are made to design scalable packet-switching architectures and to propose adapted packet scheduling schemes that simplify the complexity of the process, and guarantee high performance. Different *very well known* concepts are combined to remodel the multistage packet-switching fabric. The thesis is two-pronged:

1. Ideas have been borrowed from bufferless and fully buffered switching architectures to suggest a multistage switch with partially-buffered modules. The number of internal queues is limited and efficiently scheduling algorithms are proposed to preserve a good performance while involving less HW cost and complexity.
2. The second proposal makes the major part of this thesis. Networks-on-Chip are integrated into a multistage switching network to get over limitations of single-hop crossbars and large buffers. This design approach
 - excepts VOQs at the input stage of the multistage switch
 - simplifies the packet dispatching method
 - distributes the routing process
 - gives high performance under different traffic types

Building large high-performance packet-switches for the data center networks has been always a hot topic. Prior state-of-the-art have suggested multistage switches with bufferless fabric, costly scheduling algorithms and poor performance. Buffered fabrics with prominent number of queues claim simpler control

(than bufferless alternatives), and exhibit poor performance under skewed traffic. Unfortunately, previous solutions have limited scalability and involve costly modules or complex arbitration processes. This thesis has demonstrated that it is possible to build robust and high-performance packet-switching architectures that scale well to waving load and valency. A class of multistage NoC-based switching fabrics are described along with adopted scheduling schemes. The switches achieve high throughput levels, and guarantee a smooth variation of the average packet latency.

A survey of the switching architectures, buffering techniques, and packet scheduling algorithms is given in Chapter 2. The limitations of the single-stage switches and previous multistage switches are outlined in the context of demanding data center networks. An overview of the Networks-on-Chip was provided as well. The survey also focussed on some closely related issues to packet-switches; like congestion, flow control mechanisms, and packet reordering.

In Chapter 3, the pros and cons of the bufferless and buffered multistage switches are identified. First, a fundamental understanding of the problem of each approach is exposed. Besides, an architecture that combines design principles from both existing architectures is presented in such a way as to gain the best of both worlds. The chapter provides a detailed description the PBClos switch. Along with the partially-buffered switching architecture, two scheduling algorithms are proposed. The first scheme involves multiple, independent single-resource schedulers that work in a pipelined fashion. The algorithm yields good performance, but it mis-sequences the packets' order. The second scheduling approach calls for a central arbiter to work in between the first and second stage of the Clos-network. The algorithm is referred to as frame-based scheduling, and it distributes packets of a flow among the CMs. The frame is rebuild at the OMs, and the whole process obviates the need for costly resequencing methods.

So far, limitations of the conventional MSM and MMM switches are considered to undertake the design of a practical and cost-effective switching architecture. Chapter 4 is an inaugural step on designing multistage packet-switching architectures with NoC fabrics. The Clos-UDN switch yields many advantages as the integration of NoC modules contributes towards simpler

8. CONCLUSIONS

queuing schemes at the input modules, simpler packet dispatching algorithms, and distributed and pipelined routing. The Clos-UDN avoids the use of large crosspoint buffers, and replaces them with small distributed on-chip queues. Adopting a RR packet dispatching leads to out-of-order packet delivery. Therefore, a static configuration of the IM-CM connections is proposed. Although this reduces the three-stage switch architecture to two-stage Clos-network, the approach assures that packets of a given flow are constantly dispatched to the same CMs where they get forwarded – in-order to their output ports in the OMs.

In all of the chapters of this thesis, one or more aspects of the packet-switching design have been tackled. The simulation tests of the Clos-UDN switch have shown a bounded performance under critical traffic. Taking advantage of the NoC features, the Congestion-Aware Clos-UDN switch is elaborated in Chapter 5. The architecture – that is quoted CA Clos-UDN – is a wrapped-around Clos-network with unidirectional NoC modules. Extra inter-CM links have been devised to extend the switching space, and to allow traffic flowing between the CMs. The ultimate purpose of the CA Clos-UDN is to make the switch robust to unbalanced traffic. Instead of oblivious routing, a congestion-aware algorithm that combines information about the network status to shuffle the load among CMs is implemented. The focus is mainly on the switch performance under high loads – as they are more relevant to the context of DC networks. Simulation results have shown that the CA Clos-UDN architecture delivers good throughput and low latency even under non-uniform traffic patterns.

In Chapter 6, new insights into the concepts of the multistage switching are provided by exploiting the output-queuing scheme. The OQ Clos-UDN switch with a re-designed NoC fabric is described. The switching fabric of both Clos-UDN and CA Clos-UDN switches is made of Input-Queued mini-routers tied in a mesh fashion. As for any input queued switch, the on-grid routers need speedup to achieve high throughput. This has a direct impact on the performance of the IQ Clos-UDN switches which heavily rely on the speedup factor to offer *good* performance. Meanwhile, an OQ design is known to yield higher throughput. The application of the concept reaches its limit when the port count is large due to the high memory requirements. However, conceiving small OQ routers with

small number of I/Os is always HW feasible. Simulations have shown that the OQ Clos-UDN switch meets the design objectives (scalability, manageability, and high-performance) for small embedded memories running at reasonable speed.

Chapter 7 discusses another novel variation of the buffered Clos switches where multi-directional NoC modules are put into function. The Clos-MDN switch obviates the need for complex and costly queues at the first stage of the Clos-network. It also avoids complex and synchronized scheduling processes as well as large crosspoint buffers. These features are common to all NoC Clos switches described in this thesis. However the Clos-MDN switch presents interesting architectural details. Actually, the central MDNs have been efficiently designed to exploit the assets of NoC grids to the maximum. Virtual channels are implemented to assure a deadlock-free packet transfer across the fabric. A proactive congestion management scheme has been also proposed to regulate the micro load balancing, and to improve the switch response to tough traffic. Compared to the Clos-UDN switch, the Clos-MDN scales better and faster in size, and proves robust to load irregularity. The architecture offers a range of settings that can be tuned to sleekly achieve a given performance level for a given cost and complexity charges.

8.1 Research contributions

Figure 8.1 summarizes the contributions for which, a brief description is provided in the following part. In each chapter, a new way is proposed to re-design the three-stage Clos-packet switch. The architectural constraints, buffering, packet-scheduling, and many other related issues are considered, as they pertain to the switch design. The chapters in this thesis are organized as follows:

Chapter 3: *A partially buffered three-stage Clos-network switching architecture.* The design of the central modules comes in between a bufferless and fully buffered design. Each module has only few crosspoint queues that are logically shared among the input/output links bridging the first stage and the last stage modules. Along with the architectural design, two scheduling schemes are presented: Distributed and centralized.

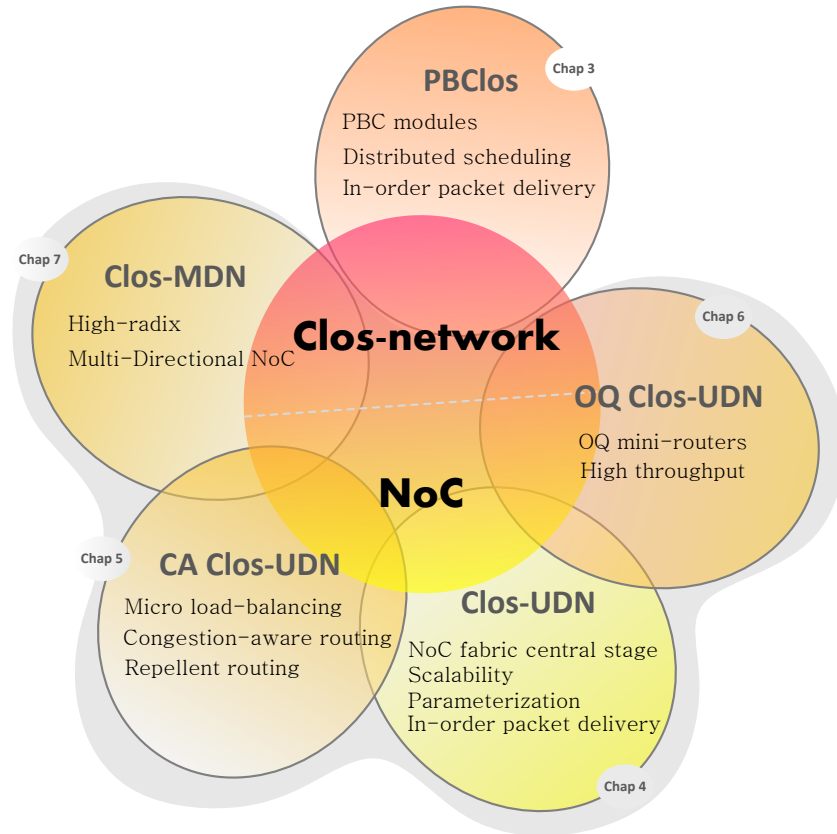


Figure 8.1: Contributions of the thesis.

Chapter 4(a): *A novel switching architecture that relies on Network-on-Chip modules.* The design mitigates the shortcomings of classical crossbar-based alternatives. It takes advantage of the NoC features to build up a switching fabric that scales well in port count, and offers high throughput levels.

Chapter 4(b): *A switch with in-order packet delivery.* Links between the first and the middle stage and links between the second and the last stages of the Clos-network are generally configured upon a matching phase. A RR dispatching scheme is implemented for the Clos-UDN switch to dynamically send packets from the switch input First In First Out (FIFO) buffers to the central NoC modules. This scheme results in packets mis-sequencing. Opting for a static configuration of the input links ensures that packets of the same

flow are constantly sent to the same CM. Inside a middle-stage module, packets of a flow get routed through the same path until the output links, and next to their corresponding output ports buffers at the third stage of the switch.

Chapter 5: *A congestion-aware load-balancing switching architecture.*

To extend the advantage of the NoC paradigm, a multistage switch with interconnected central UDN modules is designed. A congestion-aware routing algorithm that distributes the load between CMs is implemented. Under heavy unbalanced traffic loads, some CMs are likely to become more congested than others. The adaptive routing helps disperse the traffic via the interleaved inter-CM links. The approach is known as the micro load-balancing. It is to be performed at the switch level rather than the macro load-balancing in the DCN switching network which is processed by edge controllers.

Chapter 6: *A multistage switching fabric with Output-Queued NoC blocks.*

The central stage modules of the Clos-UDN have been altered and made OQ NoCs instead of input-queued NoCs. The OQ Clos-UDN switch achieves high throughput for on-chip memories that run at reasonable speed (bounded to three). The switch performance has been modelled using Markov chains and the queuing theory.

Chapter 7: *A Clos packet switch with Multi-Directional NoC modules.*

The design was proposed in an attempt to provide a highly scalable, cost-effective, and high-performance packet-switch. The potential of the NoC design is explored to suggested a three-stage Clos switch with a compact structure, programmable features, an efficient packet routing scheme that scales in size and shows robustness to traffic fluctuation.

8.2 Concluding remarks

Although it is beneficial, the curses of massive deployment still affect the field of networking. This means that new ideas can bring out interesting features and make large impact, yet they are likely to experience hard time gaining acceptance. Altering an existing network – especially large networks – to deploy new modules is technically challenging. Sometimes, it is rather complex than costly and economically infeasible. Therefore, there is more interest to plan

expandable networks using equipment that is open to co-exist with subsequent solutions. Also, great attention is reserved to design new appliances that build upon former devices. It is equally important to leave a room for future expansion purposes. This fact shows attests of the unfortunate reality that a large percentage of pertinent ideas in both academia and industry are never deployed.

8.3 Prospective work

Switches/routers are complex modules. They function at the heart of the tremendous growth and complexity of data center networks. They need be devised with care and conciliate many features, system assumptions, backward compatibility, and ever-changing requirements to do with the advent of modern applications and network protocols. So far, no cooperation was made with the industry on the design of the packet-switching architectures that are described in this thesis. However, it would have been great to spend time doing more tests to tightly examine and resolve related issues. In hindsight, there are many ways to improve and extend the current work. From within a long list, the following are interesting.

8.3.1 Future directions

In this thesis, many techniques are combined to devise new packet-switching architectures and scheduling algorithms in hopes of meeting the DC networking demand. There is – still a number of directions that need further investigation such as the followings.

1. **Out-of-order packet delivery:** This happens if no precaution is taken in designing the packet-switching architecture and the scheduling process. The NoC-based Clos switches that are described fall into the category of buffered architectures. Most of them do not grant an ordered packet delivery. Consequently, the solutions need cater for resequencing mechanisms, and allow for the associated cost and complexity of the operation.

2. **Providing performance guarantee:** The general trend is that the capacity of switches/routers goes over a buoyant growth. Future devices will have larger valency, higher data rates per port, and more sophisticated functionalities (e.g., support of QoS). Advances in the VLSI technology allow the integration of memories of judicious size, and the capacity of on-chip memories are expected to grow. This looks promising – as far as the packet-switch models presented in this thesis are concerned. It would be interesting to extend the current work, and to provide performance guarantees with feasible links and memories speed and practical buffers capacity. This must be tackled with a system-wide approach including the architectural design, scheduling, flow and congestion-control, etc.
3. **Multicast packet scheduling:** In this thesis, the unicast traffic scheduling is considered. It is all important to bring to bear schedulers that are capable of handling all types of traffic – including multicast flows, and mixture of unicast/multicast traffic.
4. **Compatibility with Software Defined Networking (SDN):** Today’s data centers require highly virtualized and scalable networks. The multi-tenancy and cloud computing are – both – driving the need for simplicity, efficiency and cost-effectiveness from the network fabric, in addition to expandability, reliability and high-performance. SDN has been introduced to enhance the resource flexibility and virtualization, and to reduce the infrastructure cost and overhead. It decouples the two fundamental functions of the network: The data plane and the control plane. An SDN-enabled DCN uses a southbound interface¹ to control the underlying switching network. So far, no special switches/routers have been claimed mandatory to deploy a software defined network. But proper extensions to the open protocols is necessary to bridge the standard and specific control interfaces. We are looking forward to testing the performance of the proposed switches in the context of SDN, and to evaluate their scalability and performance while being controlled using SDN protocols.

¹OpenFlow or an alternative protocol specification.

8.3.2 Open problems

The following are two of the major open problems in the area of networking that – hopefully – will stimulate further research and bring new pertinent solutions.

1. **Scalability of the current switching architectures:** Switches/routers at the heart of the networking substrate must – by no means – cripple the scalability potential of the DC network. This is an ineluctable prerequisite to ensure that the data center preserves its sophisticated quality of service capabilities. Usually, the complexity of implementation of any packet-switching model depends on the performance of available memories, integration process and synthesis technology. While the switching fabric architectures are designed with the intention to guarantee scalable, expandable and easily manageable modules, there are practical considerations to take into account when implementing and deploying the modules in the DCN.
2. **Unpredictability of the next DC applications:** We do not have sharp estimation on how the traffic in data centers is going to evolve. But within the next few years, new applications will come to the surface, make the traffic profile more diverse, inclined, and put demand on the switching layer. In this case the current switches/routers will not be a panacea. Most probably, modules with limited scalability features will have hard time dealing with inclined traffic. As no one can predict these applications, continuing innovation is the resolution to cater to perpetual needs, and to scale performance.



Traffic types and performance indices

A.1 Traffic types

In this thesis the performance of the multistage packet-switching architectures is assessed under different traffic types. The performance results are computed by gathering related statistics along the simulation time. Simulations are run for one million time slots, and data is collected when tenth of the simulation time has elapsed. Generally speaking, two random processes are used to define the temporal and spacial variations of a traffic pattern. The temporal variable refers to the inter-arrival times of two successive events of packet arrival. This can be deduced from the arrival frequency or the value of the input load. The spacial process is indicated by the distribution of packet arrivals over the set of switch outputs. In this appendix, the traffic scenarios that have been considered within the dissertation are described.

A.1.1 Uniform traffic

Although most of the Data center network traffic is a mixture of all types of traffic, uniform traffic is very likely to be rare. Yet, various traffic models are considered among those recommended by the switching fabric benchmarking group [195].

Bernoulli uniform traffic

This is a common test-bed traffic mostly used to evaluate the performance of packet-switches. Packets are generated in every time-slot with a given probability. In the following, ρ is the normalized input load such that all inputs are equally

loaded, and N is the switch valency. For a *Bernoulli* uniform traffic, every output port j gets $\lambda_{i,j} = \rho/N, \forall i, j \in \{1..N\}$.

Bursty uniform traffic

Bursty traffic is relevant to Data center networks. Inside the network, real traffic such as video and audio streams are segmented to form a burst or a set of bursts. So in practice, the DCN traffic is highly correlated from cell to cell, and packets tend to arrive in bursts. The bursty traffic can be simply modelled using a two-state Markov-chain: One defines the busy period when packet arrivals happen (ON), and the other defines the idle period (OFF) where no traffic arrives. Each input port is connected to a burst source that generates packets using the ON/OFF state process, and it alternates between the busy and idle periods for a geometrically distributed number of time slots. During the ON state, a cell arrives at the beginning of every time-slot, and packets of the burst are destined to the same output port. The size of the burst is generally quoted as b , and it reflects the number of packets within the frame.

A.1.2 Non-uniform traffic

Real traffic is non-uniform. This means that the distribution of the input traffic among the output ports is likely to vary. It is impossible to simulate all types of irregular traffic, however there are some representative and used non-uniform traffic models that are usually used to evaluate the packet-switch response. In this thesis, the following models are considered.

Unbalanced traffic

It is defined with reference to an unbalanced probability ω that describes how traffic arrivals are scattered among output ports of the switch. Assume an $N \times N$ switch and an input load ρ at every input port then, for each input i and output j , the traffic load is given by:

$$\lambda_{i,j} = \begin{cases} \rho(\omega + \frac{1-\omega}{N}) & \text{if } j = i \\ \rho(\frac{1-\omega}{N}) & \text{otherwise} \end{cases}$$

Note that when $\omega = 0$, the load is uniformly distributed over all outputs. When $\omega = 0.5$, then the traffic corresponds to hot-spot. This pattern is hard to schedule, and generally exposes the switch to its lowest performance. When $\omega = 1$, the traffic is said to be directional (*i.e.*, only the diagonal of the traffic matrix is not null).

Diagonal traffic

The diagonal traffic is a very skewed pattern. It is more difficult to schedule than uniform loading. It can be modelled as follows:

$$\lambda_{i,j} = \begin{cases} \frac{2\rho}{3} & \text{if } j = i \\ \frac{\rho}{3} & \text{if } j = |i + 1| \\ 0 & \text{otherwise} \end{cases}$$

The following is an example of the diagonal traffic matrix Λ for a 4×4 switch.

$$\Lambda = \frac{1}{3} \begin{pmatrix} 2\rho & \rho & 0 & 0 \\ 0 & 2\rho & \rho & 0 \\ 0 & 0 & 2\rho & \rho \\ \rho & 0 & 0 & 2\rho \end{pmatrix}$$

Log diagonal traffic

The log diagonal traffic is more balanced than the diagonal loading, but more skewed than a uniformly distributed load. It can be modelled as following: $\lambda_{i,j} = 2\lambda_{i,j+1}$, and $\sum_i \lambda_{i,j} = \rho$. For example the distribution of the load at input i across the switch outputs is given by $\lambda_{i,j} = \frac{2^{N-j}\rho}{2^N-1}$.

A.2 Performance indices

Throughout this work a time-slotted operation is assumed, where time is divided into fixed-length time slots. They are also called packet cycles, and they express the transmission time of one packet. The packet cycle equals L/R , where L is the packet length in terms of bits and R is the link rate (or port speed) in bits per second (b/s). The following metrics are considered to examine the switch performance:

A. TRAFFIC TYPES AND PERFORMANCE INDICES

- Average **throughput**: This is defined to be the average number of packets that successfully exit the switch during one packet cycle divided by the total number of switch output ports. Equivalently, this is expressed as the fraction of time the output lines are busy (output utilization). The maximum throughput is defined to be the verge input load after which the switch becomes unstable. The term instability means that the input load is higher than the throughput of the switch – in which case queues will grow unstoppably and indefinitely. The maximum throughput is also known as the saturation throughput of the switch, it and indicates the switch capacity. If the saturation throughput of a switch with a given scheduling algorithm equals to one – which is the maximum value due for a speedup of one – then, the packet scheduling algorithm is said to achieve 100% throughput (full throughput). A scheduling algorithm is said to be stable, if it provides 100% throughput, and it keeps the input buffers size bounded.
- Average **packet delay**: This reveals the average delay encountered by a packet while traversing the switch. Usually, this metric is expressed in packet cycles. Given two scheduling algorithms that both can achieve full throughput, the one with the lower average cell delay is preferable.
- Average **blocking time**: It refers to the mean time a packet spends in a buffer waiting for its next facility to become free for use. As for the average packet delay, this metric is expressed in packet cycles.

B

Upper bound on the blocking probability

The proof of the inequality 6.17 is divided into two steps: First, the availability probability in the central modules of the switch is approached, then an upper bound on the blocking probability is inferred.

Let $\{1, \dots, m\}$, be the set of output buffers of the OQ-UDN module and \mathcal{R} , the set of paths in the mesh network where each route $r \in \mathcal{R}$ is a non-empty set of output buffers connected by means of physical links. Let's define $\mathcal{R}_{r_j} \subseteq \mathcal{R}$, $r_j = \{1, \dots, m\}$, as the subset of paths that intersect in output buffer j . Assume that $\mathcal{R}_{r_j} \neq \emptyset$ and that at the steady state, an output buffer is used by at least one path in \mathcal{R} . The parameter ν_r denotes the end-to-end traffic load of route $r \in \mathcal{R}$.

Assuming that the traffic getting out of the IMs of the OQ Clos-UDN switch after the dispatching phase, is still stationary. It is worth-mentioning that although a uniform traffic is considered for this analysis, the current proof stands even for an arbitrary traffic type. At this point, no idea is provided about the traffic intensity ρ_j coming to an output buffer at a time; since it is not an input parameter like ν_r . However, this proportion of traffic is the superposition of the load carried over the previous stretch of a given path. Obviously, ρ_j depends on the availability of the upstream MRs' output buffers that may in turn depend on other factors. To simplify, an upper bound is set on ρ_j that is denoted $\check{\rho}_j$ with respect to $\{\nu_r, \check{\rho}_j \leq 1\}$.

A path is said to be blocked with a probability $B(r)$, if and only if at least one of the buffers a packet must go to on its route is not available. The blocking probability of any output buffer is an increasing function of its input traffic intensity. Diversely, the availability probability is a decreasing function of the same parameter. The blocking events might not be simply independent of

B. UPPER BOUND ON THE BLOCKING PROBABILITY

other buffers located somewhere in the mesh network which makes the situation delicate to handle mathematically [196]. Suppose that a route has χ buffers and that for any output queue, with an input load ρ_j , the blocking probability is $P_{ctr}(j)$ (evaluated in sub-section 6.3.3 of Chapter 6). A set of useful terms are introduced to prove (6.17). Consider $\alpha_r(\rho_j) = \alpha_{r_j}$, the probability that an output queue is available in a route r . It is said that a route $r \in \mathcal{R}$ is available with a probability $A(r)$, if the whole set of buffers on the path are simultaneously available. Since output buffers of MRs are not necessarily independent, the availability of the set of buffers all along a path is not always a simple product of the individual buffers availability probabilities. In other words, $A(r) \neq \prod_{j \in r} (1 - P_{ctr}(j))$, $r \in \mathcal{R}$. Note that if $r = r_j$ (a single output on the route which can happen when the number of pipeline stages of the OQ-UDN mesh $M = 1$), then $A(r) = (1 - P_{ctr}(j))$. Denote $r^{[j]}$, the j first queues on the initial segment of the route r where $j \leq |\chi|$. If the initial segment is such that $j = 0$, then the route is an empty set of buffers for which $A(\emptyset) = 1$ is defined. It is shown that with an arbitrary dependency pattern, the probability that a route $r \in \mathcal{R}$ is available always satisfies the following equation:

$$A(r) = \prod_{j=1}^{\chi} \alpha_{r_j} \left(\sum_{s \in \mathcal{R}_{r_j}} \nu_s A(s - r^{[j]}) \right); r \in \mathcal{R} \setminus \{\emptyset\} \quad (\text{B.1})$$

To prove (B.1), the term $\tilde{\mathcal{R}}$ is considered such that $\tilde{\mathcal{R}} = \mathcal{R} \setminus \{\emptyset\}$. If $\chi = 1$, then $r = r_1$ and the following equation holds

$$A(r_1) = \alpha_{r_1} \underbrace{\left(\sum_{s \in \mathcal{R}_{r_1}} \nu_s A(s - r_1) \right)}_{\psi} \quad (\text{B.2})$$

The term ψ is the sum of the traffic intensities offered on all routes that enclose the output queue r_1 multiplied by the probability that the remaining stretch of the route $s \in \mathcal{R}_{r_1}$ is available ($A(s - r_1)$). It is possible to describe ψ differently as the *route – carried* traffic load that ends up at the input of the queue r_1 , and that was previously denoted as ρ_1 . From (B.2) the following

equivalence can be derived $A(r_1) = \alpha_{r_1}(\rho_1) = \alpha_{r_1}$. Consequently, (B.1) holds for $\chi = 1$.

Next, the system of equations in (B.1) is proved to be valid for routes of length $\chi > 1$ (*i.e.*, an arbitrary number of pipeline stages such that $M > 1$). The set of events e_j , $j = 1, \dots, \chi$ is introduced to spot whenever an output queue is available with the probability $Pr(e_j)$. The probability that the whole path is not blocked can be expressed as a conditional probability in such a way that the availability of a set of outputs in the route depends on previous buffers.

$$\begin{aligned} A(r) &= Pr(e_1) \frac{Pr(e_1 e_2)}{Pr(e_1)} \cdots \frac{Pr(e_1 e_2 \dots e_\chi)}{Pr(e_1 e_2 \dots e_{\chi-1})} \\ &= Pr(e_1) \prod_{j=2}^{\chi} Pr(e_j | e_{j-1} \dots e_1) \end{aligned} \quad (\text{B.3})$$

Using the initial input traffic load of a route r and the number of buffers that a packet runs through, $A(r)$ is calculated. Clearly, the probability of availability of the route r concerns with the remaining subset of queues after the first j buffers are excluded as (B.3) shows. This means that it depends on $r - \{r_1\} - \{r_{j-1}, \dots, r_1\} = r - r^{[j]}$. Consequently, equation (B.3) can be written in a different way:

$$Pr(e_j | e_{j-1} \dots e_1) = \alpha_{r_j} \left(\sum_{s \in \mathcal{R}_{r_j}} \nu_s A(s - r^{[j]}) \right) \quad (\text{B.4})$$

Taking into account that $Pr(e_1) = A(\{r_1\}) = \alpha_{r_1}$, and using the system of equations in (B.4), the following equation (B.1) is inferred.

Being a probability, the factor $A(r - r^{[j]}) \leq 1$. Thus removing $A(r - r^{[j]})$ from the right-hand side of (B.1), should result in the following inequality:

$$A(r) \geq \prod_{j=1}^{\chi} \alpha_{r_j} \left(\sum_{s \in \mathcal{R}_{r_j}} \nu_s \right) \geq \prod_{j=1}^{\chi} \alpha_{r_j}(\check{\rho}_j) \quad (\text{B.5})$$

Where $\check{\rho}_j = \sum_{s \in \mathcal{R}_{r_j}} \nu_s$, is the traffic intensity that comes from all routes $r \in \mathcal{R}_{r_j}$,

and that intends to go to queue r_j . Finally, the general inequality $1 - \prod_{s=1}^S a(j) \leq$

B. UPPER BOUND ON THE BLOCKING PROBABILITY

$\sum_{s=1}^S (1 - a(i))$ is used and an upper bound on the blocking probability $B(r)$ is estimated.

$$B(r) \leq 1 - \prod_{j=1}^{\chi} \alpha_{r_j}(\check{\rho}_j) \leq \sum_{j=1}^{\chi} \left(\underbrace{1 - \alpha_{r_j}(\check{\rho}_j)}_{P_{ctr(j)}} \right) \quad (\text{B.6})$$

The Equation (6.17) is concluded.



Analytical modelling of the Clos-MDN switch's performance

This is part of the current work. It is to follow what is presented in Chapter 7. A step-by-step approach is adopted to elaborate an analytical model for the performance metrics of the Clos-MDN switch.

C.1 Structure of an MDN module

The structure of an MDN module is described in Chapter 7. The MDN fabric is a $\frac{k}{4} \times \frac{k}{4}$ 2-D mesh NoC that consists of a set of mini-routers, $\mathcal{V} = \{(x, y); 0 \leq x, y \leq \frac{k}{4} - 1\}$, where each $MR^1(x', y')$ is connected to its direct neighbours ($x' \pm 1$ and $y' \pm 1$) if they exist. A node is connected to its neighbouring analogous by at maximum four input and output links which are labelled *West*, *East*, *North* and *South*. Each input link has an input buffer and two virtual channels to separate the *West*→*East* (*VC0*) and *East*→*West* traffic flows (*VC1*). All input buffers have the same depth and serve in a FCFS fashion. The buffered flow control mechanism regulates the process of packet forwarding. Under this scheme, a packet is held in the input buffer of its current node until the downstream mini-router has an empty room (in the corresponding input buffer). Consequently, the MDN will not drop any packet in transit.

C.2 Assumptions

The following assumptions are made to build the analytical model.

¹To refer to a Mini-Router.

C. ANALYTICAL MODELLING OF THE CLOS-MDN SWITCH'S PERFORMANCE

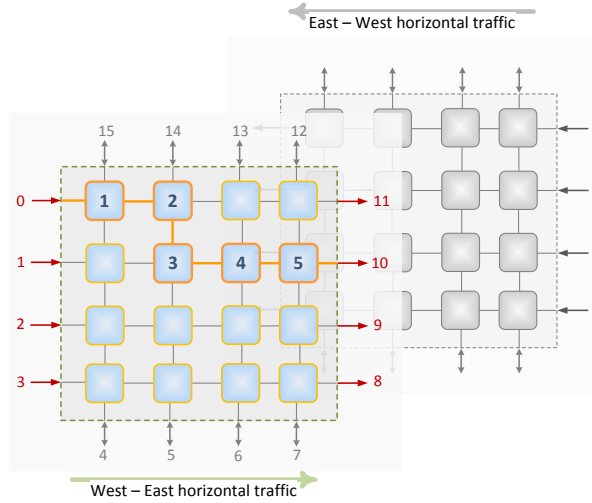


Figure C.1: The MDN can be viewed as the superposition of two UDNs with opposite horizontal traffic flow directions. It implements VCs to convey packets to their destinations based on their addresses.

- 1- Assume a Bernoulli *i.i.d* uniform traffic at the inlets of the MDN modules, in which case, no packets travel through the inter-MDN connexions.
- 2- The MDN module mesh network is organized into $\frac{k}{4}$ rows and $\frac{k}{4}$ columns with a deterministic routing (“ModuloXY” algorithm) and a store-and-forward switching mode. The on-chip routers are being labelled (x, y) which corresponds to the position of the node on the mesh grid (x row and y column).
- 3- Messages are being segmented at their injection to the Clos-MDN switch inputs. This implies that all packets in an MDN module have the same fixed size.
- 4- A packet requires one-cycle transmission time over a physical link bridging two mini-routers.
- 5- The input buffer depth is measured in units of packets, and the capacity of any input buffer is $m \cdot R$, where m is a positive integer and R is the size of a packet.
- 6- Packets are consumed immediately by the destination node.

C.3 Outline of the model

In order to characterize the MDN modules performance, information about the layout, routing and application models are essential. In the analysis, a packet is considered to be the basic/atomic unit, and the end-to-end packet latency in the switch is estimated.

C.3.1 End-to-end latency

The average packet latency is a major performance metric. Assume that the packet latency in the MDN module spans the instant a cell arrives to the input buffer of the corresponding node at the edge column of the mesh (first or last column), to the time it leaves the destination node at the opposite edge column of the mesh. A packet travels across the NoC layout such that it moves forward by one stage in the x direction or y direction at every time-slot. It visits only one input buffer in each node that belongs to its route. Next, a route $\mathcal{R}=\{r_i^{x,y}, 1 \leq i \leq \chi, 0 \leq x, y \leq \frac{k}{4} - 1\}$ is defined to be the succession of χ input buffers of mini-routers, that a packet crosses during its sojourn until exiting the NoC module. A packet is forwarded based on the routing information on its header. Upon its entry to the MDN module, the virtual channel is decided. The next hop is incrementally computed (*i.e.*, at every node), and packets of the same flow are routed through the same path in one direction using one of the two virtual channels. Having the complete picture clear, one can argue that any packet has one sole VC to use in any input buffer of a mini-router it crosses, and that the VC remains the same on all on-chip routers of the packet route.

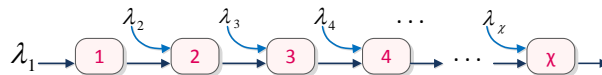


Figure C.2: A tandem of queues.

To estimate the average packet delay in the MDN, the feed-forward tandem is simplified shown in Fig.C.2. This is done in an iterative way so that each input

C. ANALYTICAL MODELLING OF THE CLOS-MDN SWITCH'S PERFORMANCE

Table C.1: *List of symbols used for the model.*

Symbol	Description
Depends on the architecture topology and routing	
$k/4$	Number of MRs per row/column
VC_1	Virtual channel for West→East horizontal flows
VC_2	Virtual channel for East→West horizontal flows
C	Capacity of inter-MRs link
(x, y)	Coordinates on a MR in the mesh, for which there is an input queue in the tandem at position i
(x', y')	Coordinates on a MR in the mesh, for which there is an input queue in the tandem at position $i - 1$
$N - 1$	Input buffer depth
χ	Number of input queues in a tandem
\mathcal{R}	Succession of χ input buffers that make up the packet route
l	Direction of an input buffer in a single MR
Depends on the application	
Pck_{src}	Packet source at the MDN grid
Pck_{dest}	Packet destination at the MDN grid
R	Packet size
T	Mean service time
$b_{x,y}^l$	Input buffer in MR(x, y) and part of the tandem
$b_{x,y}^j$	Input buffer in MR(x, y) which direction j is different from l (<i>i.e.</i> , not part of the tandem)
λ	Arrival rate at the left input buffer of a MR at an edge column of the MDN mesh
$\lambda_{x,y}^l$	Exogenous arrival rate to input buffer of direction l in MR(x, y)

queue is replaced with the delay it incurs. In [197], Neely *et al.* proposed an equivalent model for tree networks with deterministic service time. They show that for a tree network with homogeneous server rates (where the service time $T_i = T$, for all nodes), the state occupancy distribution in the final node of the tree is the same as in an equivalent model whereby all nodes before the last one are replaced with delay lines of duration T . This approach is also valid for a tandem of queues with arbitrary arrival process, as was specified in [198]. In the equivalent model depicted in Fig.C.3, the last node in the tandem is fed with a set of delayed input streams $\sum_{i=1}^{\chi} \lambda_i(t - T_i)$, where $T_i \triangleq (\chi - i)\omega_{node}$ and ω_{node} is the delay that incurs a single isolated node.

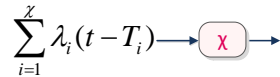


Figure C.3: *Equivalent model of the tandem of queues.*

Let $\Lambda = \sum_i \lambda_i$, be the sum rate where $\lambda_1 = \lambda$. The mean occupancy in the final node of the tandem is given by:

$$\mathbb{E}(N_\chi) = (\Lambda - \lambda_\chi) + Q\left(\sum_{i=1}^{\chi-1} \lambda_i\right) - \sum_{i=1}^{\chi-1} Q(\lambda_i) \quad (\text{C.1})$$

C.3.2 A single queue in the tandem

Note that the length of packets is fixed. Packets are served in a FCFS discipline, and the service time is a constant equal to T . Since, in addition, the input buffer depth is finite, any input buffer of a mini-router can be modelled as an M/D/1/N queue, with $N - 1$ places in the queue. The M/D/1/N queue has been long studied and solved from a computational point of view. However, there is no explicit analytical expression for the mean waiting time and the average number of customers in the system [199]. It is common practice to derive the former expressions by setting the value of the squared coefficient of the service time variation, σ^2 , to 0 in an M/G/1/N queue model [199]. In [200],

C. ANALYTICAL MODELLING OF THE CLOS-MDN SWITCH'S PERFORMANCE

Burn and Garcia inferred the expression of the coefficients of the embedded Markov chain of the M/D/1/N queue.

$$a_n = \sum_{k=0}^n \frac{(-1)^k}{k!} (n-k)^k e^{(n-k)\rho} \rho^k \quad (\text{C.2})$$

Where $\rho = \lambda_Q T$ is the utilization factor, λ_Q is the rate of a Poisson arrival process, and n is the number of packets in the M/D/1/N queue. Authors also gave an explicit expression of the mean number of customers $Q_{M/D/1/N}$, as well as the mean waiting time in the queue.

$$Q_{M/D/1/N} = N - \frac{\sum_{k=0}^{N-1} a_k}{1 + \rho a_{N-1}} \quad (\text{C.3})$$

$$\omega_{M/D/1/N} = \left(N - 1 - \frac{\sum_{k=0}^{N-1} a_k - N}{\rho a_{N-1}} \right) T \quad (\text{C.4})$$

For more details about the model of the single queue, readers might wish to refer to [200].

C.3.3 Exogenous arrival rates

In the following parts of this appendix, modelling and analysing the end-to-end packet delay in the half-MDN is considered. In this case, the traffic flows *West*→*East* using *VC0* (It makes no difference for the latency analysis in case *VC1* is in use and the logic remains the same). The first queue in the tandem is the left input buffer of the left-most column mini-router in the NoC layout. It is excursively fed by flows of packets dispatched from the Input/Output Modules of the Clos switch (Refer to Chapter 7 for the general architecture of the Clos-MDN switch). As for other queues in the tandem, the packet arrival process to a buffer $r_i^{x,y}$, can be decomposed into *endogenous* and *exogenous*. An endogenous arrival process to queue $r_i^{x,y}$ is the departure process from the previous queue in the tandem – $r_{i-1}^{x',y'}$. Exogenous arrivals can be described as the contribution of other links of the neighbour MR(x', y') to the buffer of concern – $r_i^{x,y}$. Recall that the on-grid mini-routers of the MDN module are IQ and that they connect to their direct neighbouring mini-routers with short

physical links. A packet at the HoL of an input buffer requests the output link that leads to its next hop. Upon winning the contention the intrinsic switching elements of the mini-router are activated, and the packet is routed throughout the output link to the corresponding input buffer of its next-hop.

The value of the exogenous arrival rate λ_i ($2 \leq i \leq \chi$) to each queue in the tandem is not given – as the arrival rate to the first queue $\lambda_1 = \lambda$. Therefore, it is necessary to estimate λ_i coming from nodes other than the one in the tail of the tandem.

The “MDN ModuloXY” routing

As it was mentioned earlier, information about the geometry and routing are necessary to elaborate the analytical model. The inter mini-routers links and the Round Robin arbiters embedded in each node to resolve the input contention are called NoC service elements. Consider that these elements are modelled by a latency that is considerably inferior to the queuing delay that packets experience in the input buffers. Moreover, the on-chip links are assumed to be work-conserving with a capacity $C = 1$ packet. The MDN modules use deterministic routing and a static virtual channel allocation. The effect of the routing scheme is described using a 4×4 forwarding matrix \mathcal{F} which elements $f_{e,o}$, are the routing probabilities from an input buffer e to an output link o within a single on-chip router. Rows of the forwarding matrix are the input queues of the on-chip mini-router while columns are the output links. The matrix is built by calculating $f_{e,o} = Pr(e \rightarrow o) = Pr(e) \cdot Pr(o|e)$. To this end, the inputs and outputs of a mini-router are labelled using numbers from 0 to 3 according to the next logic:

- $l = 0 \iff North$ I/O
- $l = 1 \iff South$ I/O
- $l = 2 \iff West$ I/O
- $l = 3 \iff East$ I/O

For the first virtual channel $VC0$ – used to convey *West*→*East* traffic – the forwarding matrix can be written as follows:

$$\mathcal{F} = \frac{1}{k/4} \begin{pmatrix} 0 & x & 0 & 1 \\ y & 0 & 0 & 1 \\ 1 & 1 & 0 & k \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Here is an explanation for some elements of the forwarding matrix: ① Elements of the diagonal are null since no packet is allowed to turn back through the link where it came from at the previous time-slot (*i.e.*, no cycles are allowed). ② There is no buffer at the *East* link for *VC0*, and hence no packet is sent to any output link from the *West* side. ③ The *East* output link do not receive packets from any input buffer, which explains why elements of the third column and the fourth row, are null –respectively. The matrix of mini-routers, when *VC1* is in use, can be build in the same way by calculating the probabilities of permitted moves, and zeroing prohibited ones (mind that the allowed horizontal flow would be *East*→*West*).

Estimation of the exogenous arrivals

The objective is to find an estimation for the exogenous arrival rates at queues of the tandem. It is obvious that the exogenous average arrival rate to a particular input buffer in $MR(x, y)$, is equal to sum of the average packet rates coming from all output links of the neighbouring mini-router – other than the connection link (*i.e.*, the link that connects $MR(x', y')$ to $MR(x, y)$). Unfortunately, there is no direct formula to compute the arrival rate to an input buffer. However, an iterative algorithm is proposed that can find the value of λ_i ($2 \leq i \leq \chi$) using the coordinates of the node in the NoC mesh as well as the tuple source/destination of the packet. The problem can be divided into stages at the end of each, a portion of λ_i is calculated.

Forwarding matrix reduction

The forwarding matrix can be simplified at every calculation step to keep only the elements of concern and to nullify others. This is to be done as following:

- 1- Build the matrix \mathcal{M}_b of dimension 4×4 which indicates the traffic arrival direction to the buffer b_i^l :

$$\mathcal{M}_b(u, v) = \begin{cases} 0, & u = b_l, \forall v \\ 1, & \text{otherwise} \end{cases} \quad (\text{C.5})$$

- 2- Build the output link direction matrix \mathcal{M}_{dir} of size 4×4 to specify the direction of exogenous arrival rates that need to be estimated. \mathcal{M}_{dir} is given by:

$$\mathcal{M}_{dir}(u, v) = \begin{cases} 1, & \forall u, v = b_l, \\ 0, & \text{otherwise} \end{cases} \quad (\text{C.6})$$

Define \odot to be an extraction operator where one of the two matching size matrices is binary. The result of the operator would be an element by element multiplication. For example,

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \odot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & a_{01} \\ a_{10} & 0 \end{bmatrix}$$

The reduced forwarding matrix is given by:

$$\mathcal{F}' = (\mathcal{F} \odot \mathcal{M}_b) \odot \mathcal{M}_{dir} \quad (\text{C.7})$$

The elements of only one column are different from zero while all others are null. Algorithm 4 gives details of the iterative computation of the exogenous arrival rates .

C. ANALYTICAL MODELLING OF THE CLOS-MDN SWITCH'S PERFORMANCE

Algorithm 4: Iterative exogenous arrival rates computation

Require: $\lambda, Pck_{src}, Pck_{dest}$

- 1:
 - 2: The first queue in the tandem is of index 0 and is a left buffer of a mini-router on the first column of the 2-D mesh
 - 3: The arrival rate to the first queue in the tandem is λ
 - 4:
 - 5: Using Pck_{src}, Pck_{dest} find \mathcal{R}
 - 6: start from the second queue in the tandem
 - 7:
 - 8: **for** each queue in the tandem, ($1 \leq i \leq \chi$) **do**
 - 9: Find coordinates (x, y) of the MR to which belongs the current queue
 - 10: Mark the position of the input queue, $b_{x,y}^l, l \in \{0, 1, 2, 3\}$
 - 11: Find the coordinates (x', y') of the MR to which belongs the previous queue in the tandem and that is connected to MR (x, y) via the link l
 - 12: Find the set of contributing buffers to $\lambda_{x,y}^l, \mathcal{B} = \{b_{x',y'}^j, j \neq l\}$
 - 13: **for** all contributors $b_{x',y'}^j, j \neq l$ **do**
 - 14: **do**
 - 15: Find coordinates (x'', y'') of the MR connected to MR (x', y') via j
 - 16: $\lambda_{x',y'}^j = \sum_{t \in \{0,1,2,3\}} \lambda_{x'',y''}^t \cdot f'_{t,j}$
 - 17: **while** ($x'' = 0$ and $y'' = \text{edge}$) or ($x'' = \frac{k}{4} - 1$ and $y'' = \text{edge}$)
 - 18: Exit the calculation process
 - 19: $\lambda_{x,y}^l = \sum_{j \neq l} \lambda_{x',y'}^j \cdot f_{j,l}$
 - 20: $\lambda_i = \lambda_{x,y}^l$
-

References

- [1] A. Andreyev, “Introducing data center fabric, the next-generation facebook data center network,” 2014, available at url<https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/>.
- [2] N. Farrington and A. Andreyev, “Facebook’s data center network architecture,” in *IEEE Optical Interconnects Conference*, 2013, pp. 49–50.
- [3] R. Miller, “Who has the most web servers?” 2009, available at <http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/>.
- [4] C. V. N. Index, “The zettabyte era—trends and analysis,” *Cisco white paper*, 2016.
- [5] J. Hamilton, “Overall data center costs,” 2016, available at <http://perspectives.mvdirona.com/2010/09/overall-data-center-costs/>.
- [6] C. G. C. Index, “Forecast and methodology, 2015-2020 white paper,” 2016.
- [7] C. Wu and R. Buyya, *Cloud data centers and cost modeling: A complete guide to planning, designing and building a cloud data center*. Morgan Kaufmann, 2015.
- [8] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, 2008, pp. 63–74.
- [9] J. Gripp, M. Duelk, J. E. Simsarian, A. Bhardwaj, P. Bernasconi, O. Laznicka, and M. Zirngibl, “Optical switch fabrics for ultra-high-capacity IP routers,” *Journal of Lightwave Technology*, vol. 21, no. 11, p. 2839, 2003.
- [10] N. Chrysos, C. Minkenbergh, M. Rudquist, C. Basso, and B. Vanderpool, “Scoc: High-radix switches made of bufferless Clos networks,” in *IEEE 21st International Symposium on High Performance Computer Architecture*, 2015, pp. 402–414.

REFERENCES

- [11] K. Pun and M. Hamdi, "Dispatching schemes for Clos-network switches," *Computer Networks*, vol. 44, no. 5, pp. 667–679, 2004.
- [12] Y. Xia, M. Hamdi, and H. J. Chao, "A practical large-capacity three-stage buffered Clos-network switch architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 317–328, 2016.
- [13] W. Yang, D. Dong, J. Zhao, and C. Li, "MBL: A multi-stage bufferless high-radix router," in *IEEE International Conference on Cluster Computing*, 2016, pp. 532–533.
- [14] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Computer Communications*, vol. 40, pp. 1–21, 2014.
- [15] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "Rethinking the data center networking: Architecture, network protocols, and resource sharing," *IEEE access*, vol. 2, pp. 1481–1496, 2014.
- [16] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4, 2009, pp. 51–62.
- [17] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039, 1999.
- [18] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.
- [19] N. W. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, University of California at Berkeley, 1992.
- [20] Y.-K. Park and Y.-K. Lee, "Parallel iterative matching-based cell scheduling algorithm for high-performance ATM switches," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 1, pp. 134–137, 2001.

REFERENCES

- [21] N. McKeown, “The iSLIP scheduling algorithm for input-queued switches,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.
- [22] C. J. A. Minkenberg, “On packet switch design,” Ph.D. dissertation, Eindhoven University of Technology, 2001.
- [23] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.
- [24] H. Obara and Y. Hamazumi, “Parallel contention resolution control for input queuing ATM switches,” *IET Electronics Letters*, vol. 28, no. 9, pp. 838–839, 1992.
- [25] H. Obara, “Optimum architecture for input queuing ATM switches,” *Electronics letters*, vol. 27, no. 7, pp. 555–557, 1991.
- [26] Y. Tamir and G. L. Frazier, *High-performance multi-queue buffers for VLSI communications switches*. IEEE Computer Society Press, 1988, vol. 16, no. 2.
- [27] H. Obara, S. Okamoto, and Y. Hamazumi, “Input and output queuing ATM switch architecture with spatial and temporal slot reservation control,” *Electronics letters*, vol. 28, no. 1, pp. 22–24, 1992.
- [28] M. Adisak and M. Nick, “A starvation-free algorithm for achieving 100% throughput in an input-queued switch,” in *Proceedings of the ICCCN’96 IEEE International Conference*, October, 1996, pp. 226–231.
- [29] M. K. Mehmet-Ali, M. Youssefi, and H. T. Nguyen, “The performance analysis and implementation of an input access scheme in a high-speed packet switch,” *IEEE Transactions on Communications*, vol. 42, no. 12, pp. 3189–3199, 1994.
- [30] M. G. Hluchyj and M. J. Karol, “Queueing in high-performance packet switching,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1587–1597, 1988.

REFERENCES

- [31] H. J. Chao, “Next generation routers,” *Proceedings of the IEEE*, vol. 90, no. 9, 2002.
- [32] B. Prabhakar and N. McKeown, “On the speedup required for combined input-and output-queued switching,” *Automatica*, vol. 35, no. 12, pp. 1909–1920, 1999.
- [33] M. Nabeshima, “Performance evaluation of a combined input-and crosspoint-queued switch,” *IEICE Transactions on Communications*, vol. 83, no. 3, pp. 737–741, 2000.
- [34] L. Mhamdi and M. Hamdi, “Output queued switch simulation by a one-cell-internally buffered crossbar switch,” in *IEEE GLOBECOM’03*, vol. 7, 2003, pp. 3688–3693.
- [35] Y. Shen, S. S. Panwar, and H. J. Chao, “Providing 100% throughput in a buffered crossbar switch,” in *IEEE Workshop on HPSR’07*, 2007, pp. 1–8.
- [36] L. Mhamdi, “PBC: A partially buffered crossbar packet switch,” *IEEE Transactions on Computers*, vol. 58, no. 11, pp. 1568–1581, 2009.
- [37] T. Javidi, R. Magill, and T. Hrabik, “A high-throughput scheduling algorithm for a buffered crossbar switch fabric,” in *IEEE International Conference on Communications*, vol. 5, 2001, pp. 1586–1591.
- [38] S. S. A. Omondi, “Advances in computer systems architecture 8th Asia-pacific conference,” *Proceedings Lecture Notes in Computer Science*, 2003.
- [39] F. A. Tobagi, “Fast packet switch architectures for broadband integrated services digital networks,” *Proceedings of the IEEE*, vol. 78, no. 1, pp. 133–167, 1990.
- [40] J. Garcia-Haro and A. Jajszczyk, “ATM shared-memory switching architectures,” *IEEE Network*, vol. 8, no. 4, pp. 18–26, 1994.
- [41] S. Iyer and N. McKeown, “Techniques for fast shared memory switches,” *Stanford HPNG Technical Report TR01-HPNG-081501*, 2001.

REFERENCES

- [42] E. S. H. Tse, “Switch fabric design for high performance IP routers: A survey,” *Journal of Systems Architecture*, vol. 51, no. 10, pp. 571–601, 2005.
- [43] A. K. Kloth, *Advanced router architectures*. CRC Press, 2010.
- [44] A. Prakash, S. Sharif, and A. Aziz, “An $O(\log/\sup 2/n)$ parallel algorithm for output queuing,” in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.*, vol. 3, 2002, pp. 1623–1629.
- [45] J. Networks, “M-series routers datasheets,” 2016. [Online]. Available: <http://www.juniper.net>
- [46] —, “E-series routers datasheets,” 2016. [Online]. Available: <http://www.juniper.net>
- [47] C. S. Inc., “Cisco catalyst 8500 series multiservice switch routers,” 2016. [Online]. Available: <http://www.cisco.com/>
- [48] T. M. Chen and S. S. Liu, *ATM switching systems*. Artech House, Inc., 1995.
- [49] C. S. Inc, “Cisco systems inc.” 2016, [Online; accessed 04-July-2016]. [Online]. Available: <http://www.cisco.com/c/en/us/products/switches/nexus-5000-series-switches/datasheet-listing.html>
- [50] Cisco, “Cisco 12000 series Internet router architecture: Switch fabric,” Tech. Rep., 2005.
- [51] C. S. Inc., “Cisco catalyst 6500 architecture,” Tech. Rep., 2013.
- [52] Juniper, “Qfx10000 switches: System architecture,” Tech. Rep., 2015.
- [53] L. Mhamdi, “Scheduling in high performance buffered crossbar switches,” Ph.D. dissertation, Delft University of Technology (TU Delft), 19 October 2007.

REFERENCES

- [54] L. R. Goke and G. J. Lipovski, “Banyan networks for partitioning multiprocessor systems,” in *ACM SIGARCH Computer Architecture News*, vol. 2, no. 4, 1973, pp. 21–28.
- [55] D. Vasiliadis, G. Rizos, and C. Vassilakis, “Performance analysis of blocking Banyan switches,” in *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*. Springer, 2007, pp. 107–111.
- [56] D. M. Dias and J. R. Jump, “Analysis and simulation of buffered Delta networks,” *IEEE Transactions on Computers*, vol. 100, no. 4, pp. 273–282, 1981.
- [57] M. N. Huber, E. P. Rathgeb, and T. Theimer, “Self routing Banyan networks in an ATM–environment.” in *ICCC*, 1988, pp. 167–174.
- [58] N. Zhang, “Nonblocking parallel Banyan network,” Jul. 21 1992, uS Patent 5,132,965.
- [59] D. Lawrie, “Memory-processor connection networks,” Ph.D. dissertation, University of Illinois Report UIUCDCS-R-73-557, February 1973.
- [60] F. A. Tobagi and T. C. Kwok, “The tandem Banyan switching fabric: A simple high-performance fast packet switch,” in *INFOCOM’91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s.*, 1991, pp. 1245–1253.
- [61] C. Clos, “A study of non-blocking switching networks,” *Bell System Technical Journal on*, vol. 32, no. 2, pp. 406–424, 1953.
- [62] X. Li, Z. Zhou, and M. Hamdi, “Space-memory-memory architecture for Clos-network packet switches,” in *IEEE International Conference on Communications*, vol. 2, 2005, pp. 1031–1035.
- [63] E. Oki, N. Kitsuwon, and R. Rojas-Cessa, “Analysis of space-space-space Clos-network packet switch,” in *IEEE ICCCN 2009. Proceedings of 18th International Conference*, 2009, pp. 1–6.

REFERENCES

- [64] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, “Concurrent round-robin-based dispatching schemes for Clos-network switches,” *IEEE/ACM Transactions On Networking*, vol. 10, no. 6, pp. 830–844, 2002.
- [65] P. Giaccone, “Queueing and scheduling algorithms for high performance routers,” Ph.D. dissertation, Politecnico di Torino, 2002.
- [66] K. J. Thurber, “Interconnection networks: A survey and assessment,” in *Proceedings of the ACM National Computer Conference and Exposition*, 1974, pp. 909–919.
- [67] Y. Xia and H. J. Chao, “On practical stable packet scheduling for bufferless three-stage Clos-network switches,” in *14th IEEE HPSR Conference*, 2013, pp. 7–14.
- [68] S.-Q. Zheng, A. Gumaste, and E. Lu, “A practical fast parallel routing architecture for Clos networks,” in *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, 2006, pp. 21–30.
- [69] F. M. Chiussi, J. G. Kneuer, and V. P. Kumar, “Low-cost scalable switching solutions for broadband networking: The ATLANTA architecture and chipset,” *IEEE Commun. Mag.*, vol. 35, no. 12, pp. 44–53, 1997.
- [70] J. Kleban and A. Wiczonek, “CRRD-OG: A packet dispatching algorithm with open grants for three-stage buffered Clos-network switches,” in *IEEE HPSR Conference*, 2006, pp. 6–pp.
- [71] K. Pun and M. Hamdi, “Distro: A distributed static round-robin scheduling algorithm for bufferless Clos-network switches,” in *IEEE Global Telecommunications Conference*, vol. 3, 2002, pp. 2298–2302.
- [72] F. M. Chiussi and F. A. Tobagi, “Implementation of a three-stage Banyan-based architecture with input and output buffers for large fast packet switches,” *Technical Report*, 1993.

REFERENCES

- [73] X. Ma, Y. Hu, J. Mao, J. Lan, L. Guan, and B. Zhang, “Analysis on memory-space-memory Clos packet switching network,” in *International Workshop on Advanced Parallel Processing Technologies*. Springer, 2007, pp. 209–221.
- [74] N. Chrysos and M. Katevenis, “Scheduling in non-blocking buffered three-stage switching fabrics.” in *IEEE INFOCOM*, vol. 6, 2006, pp. 1–13.
- [75] Z. Dong and R. Rojas-Cessa, “Non-blocking memory-memory-memory Clos-network packet switch,” in *34th IEEE Sarnoff Symposium*, 2011, pp. 1–5.
- [76] Z. Dong, R. Rojas-Cessa, and E. Oki, “Buffered Clos-network packet switch with per-output flow queues,” *IET Electronics letters*, vol. 47, no. 1, pp. 32–34, 2011.
- [77] ———, “Memory-memory-memory Clos-network packet switches with in-sequence service,” in *IEEE HPSR*, 2011, pp. 121–125.
- [78] Z. Dong and R. Rojas-Cessa, “Mcs: Buffered Clos-network switch with in-sequence packet forwarding,” in *35th IEEE Sarnoff Symposium*, 2012, pp. 1–6.
- [79] M. Zhang, Z. Qiu, and Y. Gao, “Space-memory-memory Clos-network switches with in-sequence service,” *IET Communications*, vol. 8, no. 16, pp. 2825–2833, 2014.
- [80] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, “High-speed switch scheduling for local-area networks,” *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, 1993.
- [81] Y. Jiang and M. Hamdi, “A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture,” in *IEEE workshop on HPSR*, 2001, pp. 407–411.
- [82] E. Oki, R. Rojas-Cessa, and H. J. Chao, “Pcrrd: A pipeline-based concurrent round-robin dispatching scheme for Clos-network switches,” in *IEEE ICC*, vol. 4, 2002, pp. 2121–2125.

REFERENCES

- [83] R. Rojas-Cessa, E. Oki, and H. J. Chao, "Maximum weight matching dispatching scheme in buffered Clos-network packet switches," in *IEEE ICC*, vol. 2, 2004, pp. 1075–1079.
- [84] C.-B. Lin and R. Rojas-Cessa, "Frame occupancy-based dispatching schemes for buffered three-stage Clos-network switches," in *13th IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Communic*, vol. 2, 2005, pp. 5–pp.
- [85] K. Pun and M. Hamdi, "Static round-robin dispatching schemes for Clos-network switches," in *Merging Optical and IP Technologies. IEEE Workshop on HPSR*, 2002, pp. 329–333.
- [86] H. J. Chao, J. Park, S. Artan, S. Jiang, and G. Zhang, "Trueway: A highly scalable multi-plane multi-stage buffered packet switch," in *IEEE HPSR*, 2005, pp. 246–253.
- [87] M. A. Henrion, "Resequencing system for a switching node," June, 30th 1992, US Patent 5,127,000.
- [88] J. Turner, "Resilient cell resequencing in Terabit routers," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 2. The University; 1998, 2003, pp. 1183–1192.
- [89] J. S. Turner, *Resequencing cells in an ATM switch*. Washington University, Department of Computer Science, WUCS-91-21, 1991.
- [90] H. J. Chao and B. Liu, *High performance switches and routers*. John Wiley & Sons, 2007.
- [91] F. M. Chiussi, D. A. Khotimsky, and S. Krishnan, "Generalized inverse multiplexing of switched ATM connections," in *IEEE GLOBECOM 1998. The Bridge to Global Integration.*, vol. 5, 1998, pp. 3134–3140.
- [92] D. A. Khotimsky, "A packet resequencing protocol for fault-tolerant multi-path transmission with non-uniform traffic splitting," in *IEEE GLOBECOM'99*, vol. 2, 1999, pp. 1283–1289.

REFERENCES

- [93] H. J. Chao and J. Park, “Flow control in a Multi-Plane Multi-Stage Buffered Packet Switch,” in *IEEE Workshop on HPSR*, 2007, pp. 1–6.
- [94] Z. Cao, Z. Wang, and E. Zegura, “Performance of Hashing-based Schemes for Internet Load Balancing,” in *Proceedings of INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, 2000, pp. 332–341.
- [95] L. Kencl and J.-Y. Le Boudec, “Adaptive load sharing for network processors,” in *Proceedings of INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2, 2002, pp. 545–554.
- [96] W. Shi, M. H. MacGregor, and P. Gburzynski, “Load balancing for parallel forwarding,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 790–801, 2005.
- [97] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic flow scheduling for data center networks.” in *NSDI*, vol. 10, 2010, pp. 19–19.
- [98] T.-Y. Tsai, Y.-L. Chung, and Z. Tsai, “Introduction to packet scheduling algorithms for communication networks,” *Communications and Networking*, pp. p263–271, 2010.
- [99] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, “Less is more: Trading a little bandwidth for ultra-low latency in the data center,” in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, pp. 253–266.
- [100] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, “pfabric: Minimal near-optimal datacenter transport,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, 2013, pp. 435–446.

REFERENCES

- [101] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, “Detail: Reducing the flow completion time tail in datacenter networks,” in *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4. ACM, 2012, pp. 139–150.
- [102] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” in *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, 1989, pp. 1–12.
- [103] G. Aggarwal, R. Motwani, D. Shah, and A. Zhu, “Switch scheduling via randomized edge coloring,” in *44th Annual IEEE Symposium on Foundations of Computer Science. Proceedings*, 2003, pp. 502–512.
- [104] J. E. Hopcroft and R. M. Karp, “An $N^{5/2}$ algorithm for maximum matchings in bipartite graphs,” *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.
- [105] E. DINIC, “Algorithm for solution of a problem of maximum flow in a network with power estimation,” in *Soviet Math. Dokl.*, vol. 11, 1970, pp. 1277–1280.
- [106] P. Giaccone, D. Shah, and B. Prabhakar, “An implementable parallel scheduler for input-queued switches,” in *IEEE Hot Interconnects 9.*, 2001, pp. 9–14.
- [107] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE transactions on automatic control*, vol. 37, no. 12, pp. 1936–1948, 1992.
- [108] S. Deb, D. Shah, and S. Shakkottai, “Fast matching algorithms for repetitive optimization: An application to switch scheduling,” in *IEEE 40th Annual Conference on Information Sciences and Systems*, 2006, pp. 1266–1271.
- [109] P. Krishna, N. S. Patel, A. Charny, and R. J. Simcoe, “On the speedup required for work-conserving crossbar switches,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1057–1066, 1999.

REFERENCES

- [110] D. N. Serpanos and P. Antoniadis, “FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2000, pp. 548–555.
- [111] Y. Li, S. Panwar, and H. J. Chao, “On the performance of a dual round-robin switch,” in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1688–1697.
- [112] N. I. Chrysos, “Request-grant scheduling for congestion elimination in multi-stage networks,” Ph.D. dissertation, University of Crete, 2006.
- [113] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, “HERMES: An infrastructure for low area overhead packet-switching networks on chip,” *INTEGRATION, the VLSI journal*, vol. 38, no. 1, pp. 69–93, 2004.
- [114] S. Arteris, “A comparison of network-on-chip and busses,” *white paper*, 2005.
- [115] L. Benini and G. De Micheli, “Networks on chips: A new soc paradigm,” *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.
- [116] W. J. Dally and B. Towles, “Route packets, not wires: On-chip interconnection networks,” in *IEEE DAC Conference*, 2001, pp. 684–689.
- [117] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, “Cost considerations in network on chip,” *INTEGRATION, the VLSI journal*, vol. 38, no. 1, pp. 19–42, 2004.
- [118] P. R. Panda, N. D. Dutt, and A. Nicolau, “On-chip vs. off-chip memory: The data partitioning problem in embedded processor-based systems,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 5, no. 3, pp. 682–704, 2000.

REFERENCES

- [119] J. Balfour and W. J. Dally, “Design tradeoffs for tiled CMP on-chip networks,” in *ACM ICS 20th International Conference*, 2006, pp. 187–198.
- [120] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. De Micheli, “Design, synthesis, and test of Networks on Chips,” *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 404–413, 2005.
- [121] Z. Shi, “Real-time communication services for networks on chip,” Ph.D. dissertation, 2009.
- [122] N. Concer, “Design and performance evaluation of network-on-chip communication protocols and architectures,” Ph.D. dissertation, alma, 2009.
- [123] K. Goossens, J. Dielissen, and A. Radulescu, “Æthereal network on chip: Concepts, architectures, and implementations,” *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.
- [124] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.
- [125] J. Duato, A. Robles, F. Silla, and R. Beivide, “A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment,” in *IEEE Proceedings on IPPS/SPDP*, 1999, pp. 240–247.
- [126] P. Kermani and L. Kleinrock, “Virtual cut-through: A new computer communication switching technique,” *Computer Networks (1976)*, vol. 3, no. 4, pp. 267–286, 1979.
- [127] A. Adriahtenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, “SPIN: A scalable, packet switched, on-chip micro-network,” in *IEEE Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 70–73.
- [128] M. Dall’Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, “Xpipes: A latency insensitive parameterized Network-on-Chip architecture for multi-processor SoCs,” in *IEEE 30th International Conference on Computer Design*, 2012, pp. 45–48.

REFERENCES

- [129] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, no. 2, pp. 62–76, 1993.
- [130] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of network-on-chip proposals," *white paper, OCP-IP*, vol. 1, p. 13, 2008.
- [131] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," in *Proceedings of the IEEE Asia and South Pacific Design Automation Conference*, 2006, pp. 36–41.
- [132] D. Starobinski, M. Karpovsky, and L. A. Zakrevski, "Application of network calculus to general topologies using turn-prohibition," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 411–421, 2003.
- [133] C. Socrates, P. Devamalar, and R. K. Sridharan, "Congestion control for packet switched networks: A survey," *International Journal of Scientific and Research Publications*, vol. 4, no. 12, p. 1, 2014.
- [134] A. Gersht and K. J. Lee, "A congestion control framework for ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1119–1130, 1991.
- [135] M. Kato, Y. Oie, M. Murata, and H. Miyahara, "Performance analysis of reactive congestion control based upon queue length threshold values," *Performance evaluation*, vol. 29, no. 2, pp. 105–125, 1997.
- [136] P. Newman, "Backward explicit congestion notification for ATM local area networks," in *GLOBECOM'93, including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston.*, 1993, pp. 719–723.
- [137] C. So-In, R. Jain, and J. Jiang, "Enhanced forward explicit congestion notification (E-FECN) scheme for datacenter Ethernet networks," in *IEEE International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 2008, pp. 542–546.

REFERENCES

- [138] N. Chrysos, L.-n. Chen, C. Kachris, and M. Katevenis, “Discharging the network from its flow control headaches: Packet drops and HOL blocking,” *IEEE/ACM Transactions on Networking*, pp. 15–28, 2016.
- [139] N. Chrysos, L. Chen, C. Minkenberg, C. Kachris, and M. Kateveni, “End-to-end congestion management for non-blocking, multi-stage switching fabrics using commodity switches,” *IBM, Res. Rep., RZ3792*, 2010.
- [140] N. Jiang, D. U. Becker, G. Michelogiannakis, and W. J. Dally, “Network congestion avoidance through speculative reservation,” in *IEEE International Symposium on High-Performance Comp Architecture*, 2012, pp. 1–12.
- [141] L. Jose, L. Yan, M. Alizadeh, G. Varghese, N. McKeown, and S. Katti, “High speed networks need proactive congestion control,” in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015, p. 14.
- [142] N. Concer, M. Petracca, and L. P. Carloni, “Distributed flit-buffer flow control for networks-on-chip,” in *Proceedings of the 6th IEEE/ACM/I-FIP international conference on Hardware/Software codesign and system synthesis*, 2008, pp. 215–220.
- [143] S. Iyer, R. R. Kompella, and N. McKeown, “Designing packet buffers for router linecards,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 705–717, 2008.
- [144] C. S. Inc, “Cisco 12000 series three-port Gigabit Ethernet line card,” 2016, [Online; accessed 23-January-2017]. [Online]. Available: http://www.cisco.com/en/US/products/hw/modules/ps2710/products_data_sheet09186a0080092085.html
- [145] S. Iyer, *Load balancing and parallelism for the Internet*. ProQuest, 2008.
- [146] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, “Variable packet size buffered crossbar (CICQ) switches,” in *IEEE International Conference on Communications*, vol. 2, 2004, pp. 1090–1096.

REFERENCES

- [147] F. M. Chiussi, A. Francini, D. A. Khotimsky, and S. Krishnan, “Feedback control in a distributed scheduling architecture,” in *IEEE GLOBECOM*, vol. 1, 2000, pp. 525–531.
- [148] P. Pappu, J. Parwatikar, J. Turner, and K. Wong, “Distributed queuing in scalable high performance routers,” in *IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, 2003, pp. 1633–1642.
- [149] F. M. Chiussi and A. Francini, “A distributed scheduling architecture for scalable packet switches,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2665–2683, 2000.
- [150] P. Pappu, J. Turner, and K. Wong, “Work-conserving distributed schedulers for Terabit routers,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 257–268, 2004.
- [151] K. Goossens, L. Mhamdi, and I. V. Senin, “Internet-router buffered crossbars based on networks on chip,” in *IEEE DSD*, 2009, pp. 365–374.
- [152] E. Bastos, E. Carara, D. Pigatto, N. Calazans, and F. Moraes, “MOTIM-A scalable architecture for Ethernet switches,” in *IEEE ISVLSI.*, 2007, pp. 451–452.
- [153] T. Karadeniz, L. Mhamdi, K. Goossens, and J. Garcia-Luna-Aceves, “Hardware design and implementation of a network-on-chip based load balancing switch fabric.” in *ReConFig.*, 2012, pp. 1–7.
- [154] L. Mhamdi, K. Goossens, and I. V. Senin, “Buffered crossbar fabrics based on networks on chip.” in *IEEE CNSR.*, 2010, pp. 74–79.
- [155] A. Bitar, J. Cassidy, N. E. Jerger, and V. Betz, “Efficient and programmable Ethernet switching with a noc-enhanced FPGA,” in *Proceedings of the 10th ACM/IEEE ANCS*, 2014, pp. 89–100.

REFERENCES

- [156] W. Zhang, L. Hou, J. Wang, S. Geng, and W. Wu, "Comparison research between XY and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," in *IEEE WRI Global Congress on Intelligent Systems.*, vol. 3, 2009, pp. 329–333.
- [157] S. D. Chawade, M. A. Gaikwad, and R. M. Patrikar, "Review of XY routing algorithm for network-on-chip architecture," *International Journal of Computer Applications*, vol. 43, no. 21, pp. 975–8887, 2012.
- [158] A. Radulescu, J. Dielissen, S. G. Pestana, O. P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, 2005.
- [159] N. Duffield, D. Chiou, B. Claise, A. Greenberg, M. Grossglauser, and J. Rexford, "A framework for packet selection and reporting," Tech. Rep., March 2009. [Online]. Available: <https://tools.ietf.org/rfc/rfc5474.txt>
- [160] S. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *IEEE INFOCOM 2005. Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2005, pp. 981–991.
- [161] W. Song, D. Edwards, J. Garside, and W. J. Bainbridge, "Area efficient asynchronous SDM routers using 2-stage Clos switches," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 1495–1500.
- [162] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *IEEE INFOCOM. 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2002, pp. 1032–1041.
- [163] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced Birkhoff-Von Neumann switches," in *IEEE HPSR Workshop*, 2001, pp. 276–280.

REFERENCES

- [164] R. Roberto and C. Lin, “Scalable two-stage Clos-network switch and module-first matching,” in *IEEE HPSR Workshop.*, 2006, pp. 6–11.
- [165] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, “CONGA: Distributed congestion-aware load balancing for datacenters,” in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 503–514.
- [166] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, “Fastpass: A centralized zero-queue datacenter network,” in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 307–318.
- [167] P. Wang and H. Xu, “EXPEDITUS: Distributed load balancing with global congestion information in data center networks,” in *Proceedings of the 2014 ACM CoNEXT on Student Workshop*, 2014, pp. 1–3.
- [168] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, “Presto: Edge-based load balancing for fast datacenter networks,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 465–478.
- [169] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, “Micro load balancing in data centers with DRILL,” in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015, p. 17.
- [170] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data Center TCP (DCTCP),” in *ACM SIGCOMM computer communication review*, vol. 40, no. 4, 2010, pp. 63–74.
- [171] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: Measurements & analysis,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009, pp. 202–208.

REFERENCES

- [172] T. Benson, A. Anand, A. Akella, and M. Zhang, “Understanding data center traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.
- [173] P. Gratz, B. Grot, and S. W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” in *IEEE 14th International Symposium on HPCA.*, 2008, pp. 203–214.
- [174] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, “EDXY—A low cost congestion-aware routing algorithm for network-on-chips,” *Journal of Systems Architecture*, vol. 56, no. 7, pp. 256–264, 2010.
- [175] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, “CATRA-congestion aware trapezoid-based routing algorithm for on-chip networks,” in *IEEE Design, Automation & Test in Europe Conference & Exhibition*, 2012, pp. 320–325.
- [176] F. Hassen and L. Mhamdi, “A multi-stage packet-switch based on noc fabrics for data center networks,” in *IEEE Globecom Workshops*, 2015, pp. 1–6.
- [177] A. Smiljanić, “Load balancing mechanisms in Clos packet switches,” in *IEEE International Conference on Communications*, vol. 4, 2004, pp. 2251–2255.
- [178] N. I. Chrysos, “Congestion management for non-blocking Clos networks,” in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2007, pp. 117–126.
- [179] F. Hassen and L. Mhamdi, “A scalable packet-switch based on output-queued nocs for data centre networks,” in *IEEE ICC*, 2016, pp. 1–6.
- [180] C. Wang, W.-H. Hu, and N. Bagherzadeh, “Congestion-aware network-on-chip router architecture,” in *IEEE 15th International Symposium on Computer Architecture and Digital Systems*, 2010, pp. 137–144.

REFERENCES

- [181] X. Chang, M. Ebrahimi, M. Daneshtalab, T. Westerlund, and J. Plosila, “Pars An efficient congestion-aware routing method for networks-on-chip,” in *IEEE 16th International Symposium on Computer Architecture and Digital Systems.*, 2012, pp. 166–171.
- [182] T. Karadeniz, A. Dabirmoghaddam, Y. Goren, and J. Garcia-Luna-Aceves, “A new approach to switch fabrics based on mini-router grids and output queueing,” in *IEEE conference on ICNC.*, 2015, pp. 308–314.
- [183] H. Elmiligi, M. El-Kharashi, and F. Gebali, “Modeling and implementation of an output-queueing router for networks-on-chips,” *Embedded Software and Systems*, pp. 241–248, 2007.
- [184] U. Y. Ogras and R. Marculescu, “Analytical router modeling for network-on-chip performance analysis,” in *IEEE Design, Automation & Test in Europe Conference & Exhibition.*, 2007, pp. 1–6.
- [185] S. Suboh, M. Bakhouya, J. Gaber, and T. El-Ghazawi, “Analytical modeling and evaluation of network-on-chip architectures,” in *IEEE International Conference on HPCS.*, 2010, pp. 615–622.
- [186] E. Fischer and G. P. Fettweis, “An accurate and scalable analytic model for round-robin arbitration in network-on-chip,” in *IEEE/ACM Seventh International Symposium on NoCS.*, 2013, pp. 1–8.
- [187] Y. Zhang, X. Dong, S. Gan, and W. Zheng, “A performance model for network-on-chip wormhole routers,” *Journal of Computers*, vol. 7, no. 1, pp. 76–84, 2012.
- [188] A. E. Kiasari, Z. Lu, and A. Jantsch, “An analytical latency model for networks-on-chip,” *IEEE Transactions on VLSI Systems.*, vol. 21, no. 1, pp. 113–123, 2013.
- [189] F. Gebali, *Computer communication networks: Analysis and design*. Northstar Digital Design, Incorporated, 2005.
- [190] ———, *Analysis of computer networks*. Springer, 2015.

REFERENCES

- [191] A.-L. Beylot and M. Becker, “Dimensioning an ATM switch based on a three-stage Clos interconnection network,” in *Annales des télécommunications*, vol. 50, no. 7-8. Springer, 1995, pp. 652–666.
- [192] L. Le and E. Hossain, “Tandem queue models with applications to QoS routing in multihop wireless networks,” *IEEE Transactions on Mobile Computing.*, vol. 7, no. 8, pp. 1025–1040, 2008.
- [193] F. Hassen and L. Mhamdi, “Congestion-aware multistage packet-switch architecture for data center networks,” to appear in IEEE GLOBECOM., 2016, pp. 1–7.
- [194] I. V. Sennin, “Design of a high-performance buffered crossbar switch fabric using network on chip,” Ph.D. dissertation, Delft University of Technology, 2008.
- [195] N. processing forum, *Switch Fabric Benchmarking Group Documents: Switch Fabric Benchmark Test Suites (NPF 2002.276.08), Performance Testing Methodology for Fabric Benchmarking (NPF 2003.213.06), Fabric Benchmarking Traffic Models, Fabric Benchmarking Performance Metrics, Switch Fabric Benchmarking Framework*, 2004. [Online]. Available: <http://www.npforum.org>
- [196] M. E. Ekpenyong and J. Isabona, “Performance modeling of blocking probability in multihop wireless networks,” *Journal of Applied Science & Engineering Technology*, vol. 4, 2011.
- [197] M. J. Neely, C. E. Rohrs, and E. Modiano, “Equivalent models for queueing analysis of deterministic service time tree networks,” *IEEE transactions on information theory*, vol. 51, no. 10, pp. 3576–3584, 2005.
- [198] M. J. Neely, “Exact queueing analysis of discrete time tandems with arbitrary arrival processes,” in *IEEE International Conference on Communications.*, vol. 4, 2004, pp. 2221–2225.

REFERENCES

- [199] J. M. Smith, “Properties and performance modelling of finite buffer M/G/1/K networks,” *Computers & Operations Research*, vol. 38, no. 4, pp. 740–754, 2011.
- [200] O. Brun and J.-M. Garcia, “Analytical solution of finite capacity M/D/1 queues,” *Journal of Applied Probability*, pp. 1092–1098, 2000.