DISS. ETH NO. _____

Scalable Flow Control for Interconnection Networks

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
FERDINAND GRAMSAMER

Dipl.-Ing., Universität Stuttgart, Germany
born 13.03.1969
citizen of
Germany

accepted on the recommendation of
Prof. Dr. Lothar Thiele, examiner
Prof. Dr. José Duato, co-examiner

2003

to my family

Acknowledgements

My achievements are deeply rooted on the input, comments, feedback, and patience from the excellent people that I was lucky to work with during the last 5 years.

First of all, I'd like to thank Professor Thiele for his willingness to coach me, his promptness, the eye-opening discussions and his valuable comments on the progress of my research. In the same light, I thank Professor Duato for his excellent observations, ideas, and creativity that inspired my dissertation. I am grateful to his stimulating feedback.

I am happy to have had the chance to perform my studies in such a renowned place as the IBM Research Laboratory in Rüschlikon. This once in a life-time opportunity was enriched by an outstanding work-environment and most proficient colleagues. In particular, I'd like to express my sincere appreciation to Ronald Luijten for his supervision, help, and his capability in filtering out the essence of things and to Mitch Gusat for his superb advice, vision and experience on interconnection networks that he transmitted to me.

I am indebted to all the past and current colleagues of the PRIZMA team, in particular to Cyriel Minkenberg for his helpful feedback, and sharing his profound knowledge on switch architectures and modelling with me. Thanks to Francois Abel, Wolfgang Denzel, Mark Verhappen, Markus Sing, and Silvio Dragone for friendship, and countless fruitful work- and non-work related discussions. Also thanks to Ilias Iliadis for his useful comments that have helped improve Chapter 4.

Thanks to Peter Klett and Thomas Kretschmer from the Intellectual Property Department for the fun and the good cooperation, to Charlotte Bolliger for proofreading and correcting all my publications.

Last but not least I am thankful to my wife for her patience, support and love that she gave my throughout.

Abstract

Performance of most digital systems is constrained by their communication and interconnection, not by their processing power. The technological progress makes memories and logic smaller, faster, and less expensive, whereas the corresponding necessary pin and wiring density scales at a slower rate. The clock rate of modern processors outpaces the frequency of communication between components. These factors combined make interconnection the key factor in the design of future digital systems.

Due to the extraordinary growth of the Internet, communication components such as routers and switches grow in size as they connect more computing resources at higher speeds. Unlike previous generations of networking components, these devices are no longer feasible on a single chip, board, or rack. It requires a new kind of system-level communication among multiple racks. Such large communication components are akin to multiprocessor systems with similar requirements:

- loss-less communication based on fix-sized packets
- efficient bandwidth utilization
- scalable in terms of attachable networking nodes

Hop-by-hop closed-loop flow control with selective backpressure is hereby often used to meet above requirements. The following issues have not yet been dealt with:

- Although a variety of such flow control schemes exist, no scheme triggers the return of flow control information based on the transmission- and reception-sided buffer state.
- The system-level flow control has been looked at for negligible round-trip time delays, whereas the impact of long links on resource requirements and performance has not been analyzed.
- The control overhead has not been minimized.

The above issues are addressed in this thesis and the major contributions in the research area of packet switches and interconnection networks are:

- The scheduling of flow control has been introduced. It allows to minimize the overhead necessary in feed-back control systems. The resulting scheme is distributed, autonomous, scalable and achieves load-balancing to optimize the utilization of resources.
- Various strategies are explored for this flow control scheme by exhaustive simulation for a wide set of system and traffic parameters. The exploration focuses on the resolution of output contention.

- The trade-off between implementational complexity and flow control overhead for buffered packet switches has been quantified for all combinations of relevant schemes.
- A multiplicity of highly loaded parallel flows through a buffered switch result in high aggregate throughput. This is a classical means to evaluate the quality of switches. Depending on the output queue service, throughput and delay for single flows may depend on the existence of other flows. The power metric is introduced as quality criteria to give a more distinguished view.

Kurzfassung

Die Leistungsfähigkeit der meisten digitalen Systeme wird bestimmt durch ihre Kommunikation und Verknüpfung und nicht etwa durch ihre Rechenleistung. Der technologische Fortschritt bewirkt, dass Speicherelemente und Logik kleiner, schneller und billiger werden, während die dazu nötige Pin- und Verkablungsdichte langsamer zunimmt. Die Kommunikationrate zwischen Komponenten kann nicht mehr Schritt halten mit der Entwicklung der Taktraten moderner Prozessoren. All diese Faktoren zeigen, dass Verbindung der Schlüsselfaktor im Entwurf zukünftiger digitaler Systeme ist.

Wegen des ausserordentlichen Wachstums des Internets, wachsen Kommunikationskomponenten, wie Verbindungsknoten und Vermittlungsstellen, in ihrer Grösse, da sie immer mehr Rechenpunkte mit immer höherer Geschwindigkeit verbinden. Anders als vorhergehende Generationen von Netzwerkkomponenten, sind sie nun nicht mehr auf einem Chip, Karte, oder gar einem Gestell unterzubringen. Es wird eine neuartige Kommunikation auf Systemebene zwischen den einzelnen Gestellen verlangt, die nun für das Gesamtsystem notwendig sind. Solche grossen Kommunikationskomponenten ähneln Multiprozessorsystemen, die auch ähnliche Anforderungen haben:

- verlustfreie Kommunikation basierend auf Packeten gleicher Länge
- effiziente Bandbreiteausnutzung
- skalierbar für zusätzliche Netzwerkknoten

Geschlossene Flusskontrolle zwischen benachbarten Knoten mit selektiver Rückkopplung wird hierbei häufig verwendet, um oben genannte Anforderungen zu erfüllen. Die folgenden Themen sind is jetzt noch nicht behandelt worden:

- Obwohl eine Vielzahl von Flusskontrollansätzen existiert, retourniert keiner sein Informationen basierend auf dem Zustand sowohl der Übertragungs- als auch der Empfangsseite.
- Die Flusskontrolle auf Systemebene wurde bis anhin für vernachlässigbare Umlaufzeiten analysiert, während der Einfluss von grossen Umlaufzeiten auf Resourcen und Leistungsfähigkeit nicht angeschaut wurde.
- Der Kontrollüberhang wurde nicht minimiert.

Die oben genannten Themen werden in dieser Arbeit berücksichtigt, so dass sich die Hauptbeiträge im Forschungsgebiet der Packetvermittlungsstellen und Verbindungsnetzwerke folgendermassen zusammenfassen lassen:

- Eine Regelung zum Abruf der Flusskontrolle wurde eingeführt. Sie erlaubt es den Überhang, der notwendig in Rückkopplungssystemen ist, zu minimieren. Der daraus resultierende Ansatz ist verteilt, autonom, skalierbar und gleicht die vorhandenen Lasten so aus, dass der Gebrauch der Resourcen optimiert werden kann.

- Verschiedne Strategien werden hierfür mittels Simulation unter Berücksichtigung eines weiten Spektrums von System- und Verkehrsparametern ergründet. Das Interesse liegt hierbei bei der Auflösung von ausgangsseitigem Stau.
- Der Kompromiss zwischen Implementierungskomplexität und Flusskontrollüberhang für gepufferte Paketvermittlungsstellen wurde quantifiziert für alle Kombinationen relevanter Methoden.
- Eine Vielfalt von stark belasteten parallelen Flüssen durch die gepufferte Vermittlungsstelle resultiert in hohen Gesamtdurchsatz. Dies is der klassische Ansatz, um die Qualität von Vermittlungsstellen zu bestimmen. Abhängig davon wie die Ausgangswarteschlangen bedient werden, hängt der Durchsatz und die Durchlaufzeit von der Existenz anderer Flüsse ab. Das “Powermass” wird als Qualitätskriterium eingeführt, um sich ein differenzierteres Bild machen zu können.

Contents

List of Figures

List of Tables

List of Symbols

ρ throughput, intensity

B buffer size

BS average burst size with geometrically distributed burst lengths

I flow control information

L load; the average traffic load offered to the switch fabric

M cross-point memory or memory size of the buffered switch; M and B are used interchangeably

N number of input or output ports of a switch fabric

P packet length expressed in bits/bytes

RTT round-trip time; typically expressed in packet cycles

tu a time unit expressed in packet cycles

Abbreviations

ATM	Asynchronous Transfer Mode
ARQ	Automatic Repeat Request
BAT	Bandwidth Allocation Technology
BGM	Bipartite Graph Matching
CBR	Constant Bit Rate
CIOQ	Combined-Input Output-Queued
CICQ	Combined-Input Crosspoint-queued
CMOS	Complementary Metal-Oxide Semiconductor
DRR	Dual Round Robin
ECC	Error Correction Code
FC	Flow Control
FCD	Flow Control Domain
FCSD	Flow Control Subdomain
FFT	Fast Fourier Transformation
FIFO	First-in First-out
Gb/s	Gigabit per second
HOF	Highest Occupancy First
HOL	Head-of-Line
HDLC	High-Level Data Link Control
IBM	International Business Machines
IA	Input Adapter
ISO	International Standard Organization
LAN	Local Area Network
LOF	Lowest Occupancy First

LOF-CLIQ	LOF Communicated Length of Input Queues
LT	Line Termination
LU	is not an abbreviation, but relates to the factorization of a matrix into submatrices L and U
MAN	Metropolitan Area Network
MESI	Modified Exclusive Share Invalidate
MP	Multi Processors
NIA	Network Interface Adapter
NP	Network Processor
NUMA	Non-Uniform Memory Access
OA	Output Adapter
OSI	Open Systems Interconnection
OC	Optical Carrier
OCF	Oldest Cell First
OQS	Output Queue Scheduler
RED	Random Early Discard
RR	Round Robin
RSIM	Rice Simulator for Instruction-level parallelism Multiprocessors
SA	Switch Adapter
SAN	Storage Area Network
SES	Scientific and Engineering Software
SF	Switch Fabric
SP	Scalable POWERparallel
SPLASH	Stanford Parallel Applications for Shared Memory
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SMP	Shared Memory Processor
VBR	Virtual Bit Rate
VOQ	Virtual Output Queueing
VLSI	Very Large Scale Integration

WAN Wide Area Network

WFQ Weighted Fair Queueing

WRR Weighted Round Robin

Chapter 1

Introduction

1.1 Relationship to a General Problem

Having the right information at the right time has always been a great advantage. Consider for instance someone driving in a car in a region unknown to him. If the street signs were to show him all possible options towards the destination chosen, he would be overwhelmed by the wealth of information. On the contrary, it would take him much longer to arrive at his destination as if the signs would just signal one option. When looking for a parking lot in a big city, the right information at the right time can really be useful. The driver has a parking garage in mind, but does not know whether it is full. Well, he just tries and it might turn out to be no problem, but he does not know before hand. If, however, the garage is full, it not only contributed significantly to his searching time, but also increased the overall utilization of the streets. If there are signs at major points in the city that inform the drivers about the parking status of the various garages, a driver will know in advance whether it is worth going where he initially wanted to park. With such a parking information system, which is by the way being employed in major cities in Europe, the time to park one's car can be reduced. This is a great advantage for the individual driver, as he is faster in finding a parking space. It is also an advantage for the overall traffic situation as a whole, because drivers will not contribute to traffic congestion, as they would if their parking aim was unsuccessful and they had to look for new opportunities.

This observation is the basic principle of the optimization effort in the design of switch fabrics carried out in this thesis. It will be applied for the flow-control mechanism that is used between switch adapters and the switch core.

1.2 Status Quo in Switching

As switching is being realized in hardware it is exposed to the constant progress in technological development. Moreover, switch design has to meet market and customer requirements.

The market requires not only higher port speeds, but also more ports. The customer requires more distinctions of traffic classes, that is more knowledge about what kind of packets are being switched, to differentiate classes of service.

A switching subsystem consists of adapters, typically one per switch port, which need to be interconnected by a switch core. As the transistor density scales with roughly 1.5 and the chip area with 1.1 to 1.25 per year, the number of transistors per chip increases with 1.6 to 1.8 per year. This is commonly known as "Moore's Law". With the increase in chip density, there is a corresponding decline in the cost per transistor. As a consequence larger buffers become

affordable nowadays. Moreover, computational power, i.e. the logic on a chip, becomes cheaper over time. With current CMOS technology, the problem is to move data on and off the chip and with current interconnection technology across boards and card edges. We refer to this broadly as the problem of interconnection technology which has much more facets, such as size and pitch of card connectors, speed and supported distance of backplane interconnection systems, card and rack form factors, layers of metal realizing the chip internal wiring, and as the packaging problem to move the data on and off the chip. Current electrical backplane technology allows transmission rates of 2.5 Gb/s. For Terabit switches this means thousands of wires, and possibly hundreds of adapters. Such a large switching subsystem clearly no longer fits into a single switch rack, i.e. switch and adapter cards that are connected through a single backplane. A solution are multi-rack systems that are connected by cables, inherently increasing the round-trip time of intra-fabric control loops. Looking back over the past years, the relative cost of interconnect versus switch function has gone up, indicating a shift in optimization criteria. This is the underlying switch-architectural problem domain of this thesis. The interconnection technology is therefore a bottleneck and major contributor to switch cost. Therefore, the switch fabric internal control overhead due to a specific switch architecture design has to be optimized.

As switches are not only used in communication environments, but also in storage area networks, and multiprocessor systems including clusters of PC/workstations, the same concepts and components should be re-usable in these systems. Moreover, as switches also serve different market segments, such as high-end or low-end devices, one architecture should scale to fit all segments. To this end, this thesis was conducted to merge the switch design of communication and multiprocessor environments, in particular because the switch fabrics of communication systems are no longer realized on a single physical entity, but across a multiplicity of physically distant components. This development of switches in communication environments approaches it to a multiprocessor system, and henceforth leads to an overlapping problem domain.

Single-stage switch fabrics are often realized in a combined input- and output-queued architecture. This architecture comes in two flavors, one with a buffered and the other with a bufferless switch. Typically, both apply input queueing per output, commonly known as virtual output queueing. Both have certain shortcomings that we want to address in this thesis.

The bufferless switch fabric has a centralized controller, which schedules packets at all input adapters according to traffic and output availability. It applies scheduling algorithms that are tuned for high aggregate throughput [90]. These scheduling algorithms are challenging to implement as a global state of all adapters and outputs are necessary [96], which requires a high degree of interconnectivity. This clearly is a bottleneck when scaling to larger fabric sizes. However, the problem is not only the interconnection of VOQs and a centralized controller, but also the implementation of an algorithm that is capable of finding a best match of N inputs and outputs in a limited number of packet cycles, such as parallel, iterative matching algorithms.

The buffered switch fabric is an architecture that yields a high aggregate throughput. It typically employs a shared memory architecture that is limited by memory access bandwidth [34] or the size of input and output routers to packet storing locations [93]. Moreover, the amount of flow-control information to convey is large.

If not properly addressed, these two architectural choices leave little room for supporting different traffic classes. An architecture that seems to be very attractive for supporting quality-of-service (QoS) by design is a cross-point buffered switch architecture. The drawback of such an architecture is the large amount of buffers required as it scales on the order of $O(N^2)$, N being the number of switch ports. However, with the technological progress mentioned earlier, this disadvantage increasingly looses importance as compared to the other two architectures. It was only recently that this type of architecture received more attention [47, 46, 49, 48]. We

believe that time has come to investigate this type of architecture. A recent publication showed that such an architecture is also feasible for large round-trip times [41]. Because of the high cost of buffers, researchers refrained from investigating such a switch architecture. This thesis will analyze this architecture, in particular considering large round-trip times.

1.3 Open Problems

In view of the market demands for ever higher bandwidth and more ports and because of the interconnection bottleneck, a high-speed switch fabric will be realized in the foreseeable future using long links. However, the question of the impact that long links have on performance and resource consumption has not yet been addressed appropriately. For instance in [11], long links and flow control were analyzed, but under the aspect of a single link as part of an irregular topology. In our work, we address similar aspects from the perspective of a single link as part of a switch fabric, which is an aggregation point of links. We additionally address issues of flow-control overhead.

Owing to the fact that in the past a switch had been feasible on a single rack, the issue of flow-control bandwidth was rarely considered. For instance in [17], flow-control channel limitations have been considered for TCP/IP (Transmission Control Protocol/Internet Protocol), which is an end-to-end flow-control protocol. While the authors describe the phenomena of acknowledgment congestion, which relates to the congestion of flow-control events in this thesis, and its impact on performance of one and multiple TCP connections, our research is more strongly focused on, but not limited to, a link-level flow-control scheme. We offer flow-control scheduling as a solution to reverse path congestion. We believe that flow-control scheduling can be applied without limitations to scheduling acknowledgements in an end-to-end flow-control protocol, such as TCP/IP. However, this would be beyond the scope of this thesis.

Although important, flow control is an issue that has attracted less attention in recent years. It has numerous facets that are complex and often confusing. For instance in [23], a comprehensive survey of flow control was given. Since this publication, which dates back to 1979, much has changed and some additions to that survey, such as an evaluation of flow control complexity and switch-fabric-internal flow control schemes, are necessary. There has been a come-back of flow-control issues in the early 90's. The discussion was led by [8, 10] who made credit flow-control popular. The grant flow-control scheme was carried on for instance by [28] with a significant contribution on buffer efficiency. We deliver a framework that allows to catalogue the different schemes.

In the past, the main focus of research in switch architectures was on improving the aggregate throughput of the fabric [90], [37]. It is only recently that more attention is being paid to the aspect of QoS support [48], [60], and [61]. Even though scheduling algorithms are important, scheduling yield can only be as good as the underlying architecture. The relationship between switch architecture and QoS support has not been considered thoroughly.

1.4 What Makes the Problems Difficult?

Flow control has to provide correctness and assure work-conservation. These requirements have to be fulfilled, and set the framework of any flow control scheme. Often, detailed issues of flow control were intentionally ignored when dealing with switch performance so that the influence

of factors such as buffer size, input and output-queueing, scheduling, packet sizes, traffic behavior, or switch dimension is better understood. When taking also link-length into account, the parameter scope widens, resulting in an even more complicated analysis.

As flow control is considered a solved problem, it is difficult to convince colleagues and researchers of the necessity of reconsidering flow control.

Provided an infinite flow control channel, it is easy to achieve, with the switch fabric assumed here, good performance results. However, we try to minimize the flow control overhead and maintain the good performance. It is not obvious how to achieve this, in particular when exposing the switch fabric to communication and multiprocessor environments and their typical traffic behavior.

Issues of QoS have been reduced to simple priorities in the past. Already here the issues are very complex. We will attempt to provide a foundation by means of an appropriate switch architecture to have better support of QoS-related aspects that go beyond priorities such as bandwidth and delay guarantees.

Finally, all ideas ultimately have to be feasible, as the switch will be built in real hardware. We refer to [41] for a comprehensive feasibility study of the switch architecture utilized in this thesis.

1.5 Contributions of this Thesis

This thesis starts with a comprehensive survey of flow control schemes and their application in buffered switches. It introduces a classification of flow control schemes that embeds relevant existing schemes. Flow control complexity is introduced as an abstraction of the chip area consumed, which allows a quantification of the trade-off between implementational cost and flow-control bandwidth, i.e. flow control overhead. With flow-control scheduling, a mechanism is introduced that achieves robust switch performance for varying traffic characteristics and link lengths at low flow control overhead. This makes the scheme attractive to be used in communication as well as in multiprocessors switches, and allows the switch dimensions to be scaled. Furthermore, flow control scheduling is a reusable building block to realize multistage fabrics. With flow control scheduling, a simple and easy-to-understand principle provided by mother nature has been realized in technology. It is the principle of filtering information, which is, for instance, applied in the human body between what the eye sees and what the brain grasps from theses images. In Section 1.1 we gave examples from everyday life in which these principles are also useful. We therefore see a certain generality in our results that could make the scheme attractive for instance for TCP/IP by scheduling the reverse channel, which carries the acknowledgements of typically different connections. We provide reasons for and insight into the necessity of the output-queue organization of switches to support different QoS aspects. These aspects relate to communication as well as multiprocessor switches.

Beyond the scope of this thesis are multi-stage switch fabrics, end-to-end flow-control schemes, and multicast or multi-class traffic.

1.6 Organization of the Thesis

This dissertation is structured as follows: Chapter 1 (this chapter), provides a general overview of this thesis by relating it to an everyday problem, summarizing the current state of the art in switch architecture, pointing out shortcomings and briefly discussing the solution this thesis tries to give. Chapter 2 outlines the problem domain. Here, we abstract the two target switching environments of communication and multiprocessor systems to derive a unified model. We give reasons for the abstractions and assumptions made in this work. As we perform our analysis mainly by simulations, we also present the simulation technique and the traffic patterns applied to the model. After having prepared the problem domain, we state the objectives of this dissertation in Chapter 3. Flow control is introduced in Chapter 4. Here, we present all terms and issues related to flow controlling a single link and classify existing flow control schemes. In Chapter 5 we expand the flow control design problem to N dimensions at either side of the link, and give examples of the resulting arrangements in networking in general and in switching in particular. Based on this discussion, we show in Chapter 6 that for switch design it is advantageous to view the switch as an aggregation point of autonomous flow control domains and propose flow-control scheduling as a solution for reducing flow-control bandwidth while maintaining switch performance. In Chapter 7, we propose various flow-control scheduling strategies and explain their mechanisms. We do a performance analysis by means of simulation to evaluate the behavior of flow control scheduling and the proposed strategies in comparison to an unscheduled flow control channel of unrestricted bandwidth. Furthermore, we provide substance to the discussion on how a switch architecture should be constructed to be able to optimally support QoS aspects. The discussions are two-fold, one for a communication and another one for a multiprocessor environment. Finally, Chapter 8 summarizes the conclusions we arrived at in the course of the work presented in this dissertation.

Chapter 2

System View

In the introduction, Chap. 1, we outlined that we want to achieve convergence in switches for communication systems, storage area networks and in switches for multiprocessor environments, including clusters of PCs/workstations. For convergence a unified model is necessary. We will derive such a model in Sec. 2.3. This model will be used throughout this thesis to investigate the problems, described in Chap. 3, and do analysis by means of simulation.

Before arriving there, we need to outline a typical example of each system, which is done in Sec. 2.1 for communication and in Sec. 2.2 for multiprocessor switches. We will make abstractions to each system to find a valid model. It is explained why these abstractions do not limit us in the results that will be obtained.

After having found a unified model, we will present our chosen simulation technique in Sec. 2.4 and present the traffic that will be exercised on the model in Sec. 2.5.

A list of definitions that will be used throughout this thesis is presented in Appendix. A. The words that are defined here are written in italic upon their first encounter in the text.

2.1 Communication Systems Example

A switch fabric (SF) is exposed to all type of networks and network protocols. Therefore, we start our considerations at a very high level to better understand how a SF is typically embedded in communication systems.

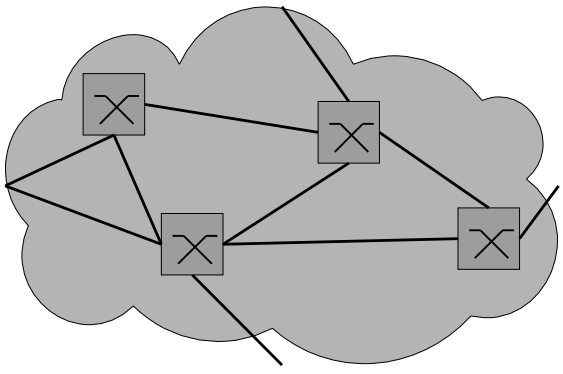


Figure 2.1: A Communication Systems Network

Typically, communication systems are drawn by a cloud representing a network, be it a Local Area Network (LAN), Metropolitan Area Network (MAN), or Wide Area Network (WAN), that runs any type of network protocol, as shown in Fig. 2.1.

The network cloud consists of networking nodes, represented by the boxes in above Figure. The edges interconnecting those boxes are links typically running at optical carrier (OC) speeds, for instance OC-192 (10Gb/s) or OC-768 (40Gb/s). Each networking node is an aggregation point of edges, which interconnects incoming links with outgoing links. Its functions are briefly described as follows:

- **line termination:** allows to physically attach a multiplicity of transmission systems to the node and provides framing functionality;

- **protocol processing:** provides the intelligence and processing power to analyze packet headers, look up, compute and update routing tables, classify packets into predefined service classes, perform other control information and (often complex) rules, and to provide queueing and policing of packets;

- **switching:** provides high-speed, non-blocking [86] interconnection of the various functional units of the networking node.

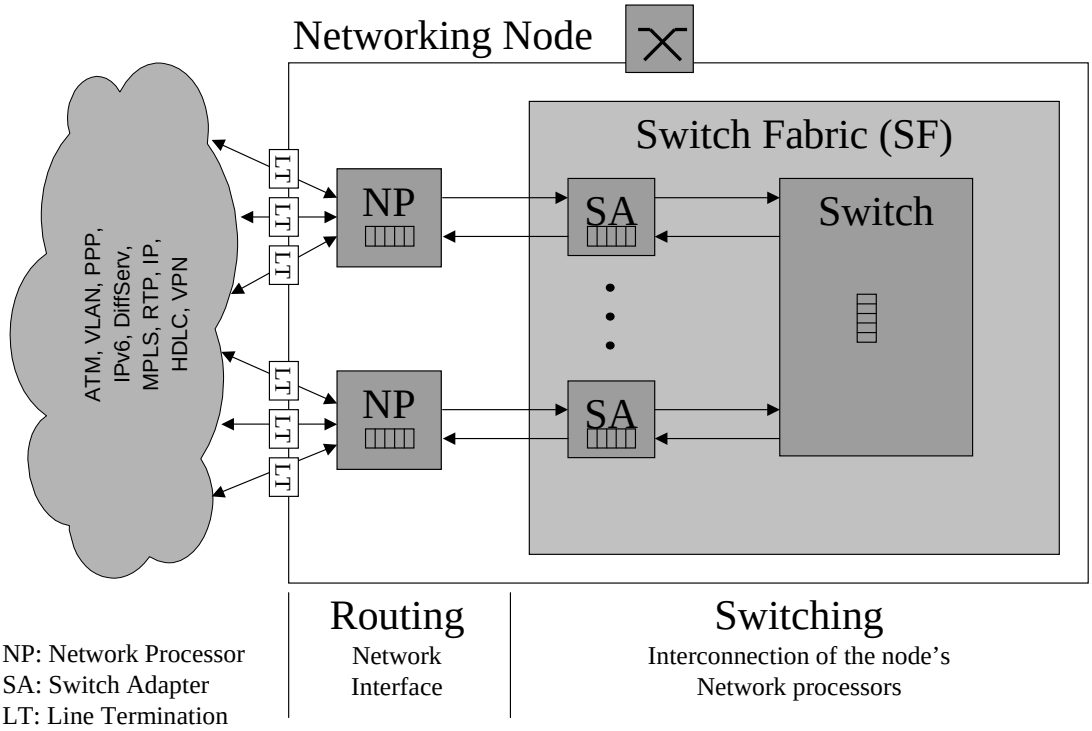


Figure 2.2: Inside a Networking Node

These functions are typically split across logical entities, such as a line termination (LT), network processor (NP) and a SF, which is shown in Fig. 2.2.

For further discussion, we define a queueing point to be a point where a packet can be parsed, checked, modified, or scheduled and experiences queueing time. Queueing time is defined for instance in [97] and is composed of waiting time and service time. Waiting time is the time interval between arrival time and the instant the service begins. Service time is the time required to perform a set or a subset of the functions listed above on the packet. Queueing points are symbolized by a finite queue.

We also define a *frame* to be a data unit of variable length.

2.1.1 Network Processor (NP)

The NP, sometimes also referred to as Traffic Manager (TM), is a queueing point both for ingress and egress. As a functional unit it performs protocol processing as described above. Therefore, it is able to interface to different type of network protocols, such as ATM, VLAN, IPv6, DiffServ, or RTP just to name a few. The ingress NP connects to the input side, while the egress NP connects to the output side of the SF. The ingress NP receives frames of data from the line interface and provides the SF with routing and possibly quality of service (QoS) information, i.e. to let the SF know to which egress NP to forward a frame to under what constrains. Therefore, the network protocol is not seen at the SF. The received data at the egress NP is queued again before being transmitted to the LT and on the output link. It may incorporate sophisticated scheduling algorithms to support priorities and guarantees. Since our interest is the SF, we can already make the following statements:

1. In order to be protocol independent, we allow connection-less, as well as connection-oriented operation.

2. The type of network protocol is not relevant to our work - also due to statement 1. As a conclusion, our SF is applicable to all communication network environments as outlined in Fig. 2.1.

3. If a protocol requires QoS provisioning, the SF will support it.

2.1.2 Switch Fabric (SF)

For implementational complexity reasons, it is a well-accepted design technique to use fixed-size packets for packet switches at the hardware level [51, 96]. Throughout this dissertation, we will use the term *packet* for fixed-sized data units. We therefore a assume our SF to operate based on packets.

The SF performs the following two main functions:

Routing: Each packet must be directed independently from its arriving input to the destination output according to the routing information provided by the NP. We assume the routing (and QoS information) given from the NP to be incorporated into the packet, namely into the packet header. For this reason packets are referred to as being *self-routing*.

Queueing: Simultaneous arrival of packets at different inputs destined to the same output causes *output contention*, assuming that only one packet may leave an output per packet cycle. Without dropping packets that lose contention, queueing is necessary. Queueing is therefore a means to resolve output contention. In order to be able to maintain the sequence of flows, a flow being a stream of packets between arbitrary pairs of inputs and outputs of a given service class, all queues obey the first-in first-out (FIFO) service discipline. This is a requirement to be protocol independent and henceforth to also support connection-oriented traffic or service classes. Obviously, this supports connection-less operated protocols as well.

The placement of the queueing function determines the class of packet switch architecture. Several basic approaches for packet queueing in packet switches are known [87]. The two classical architectures are based on input or output queueing. A switch belongs to the input queue class of switches, if its queueing function is performed before the routing function. If it is done after routing, it is attributed to the output queue class of switches [96]. While the

input queueing approach suffers from the head-of-line (HOL) blocking phenomenon limiting the maximum throughput to 58.6% [86], but has the advantage of low complexity and cost, the output queueing approach delivers optimal performance at high implementational complexity of the output queues. They must provide write access to a multiplicity of input ports. This can be achieved by operating a single queue at sufficiently higher speed or by higher paralleled implementation. Both approaches are complex and costly [34]. A reasonable compromise was proposed by [34], that allows output queues to only operate at a fraction of the required speed-up or parallelism. It is a so-called combined input output queued (CIOQ) switch architecture. To avoid packet loss, it employs a flow-control (FC) mechanism between output and input queues. The grand part of the buffer space can economically be implemented at the input, while only a small fraction is required for performance reasons at the output. The authors of [34] point out that the input queues do not necessarily have to be part of the same physical device that realizes the routing function and output queueing.

We adopt this approach in this dissertation and split therefore the SF into a switch adapter (SA) that realizes the input queues and a switch core, or simply a switch that incorporates the routing matrix and the output queues. We consider single stage fabrics only, but like to point out that our solutions shown in this dissertations seem to be very attractive for multistage fabrics.

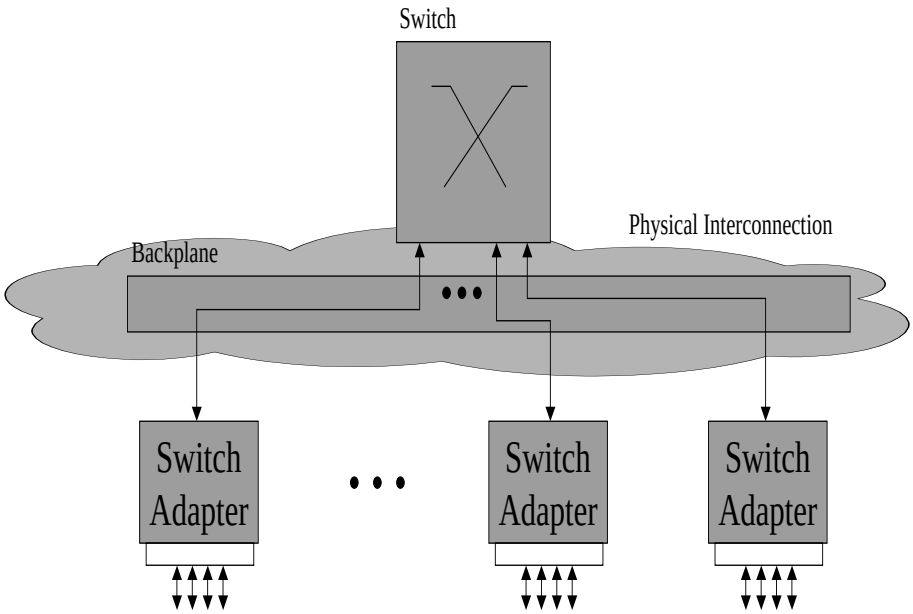


Figure 2.3: Previous Generation of Switch Fabrics

SAs are interconnected to the switch via links that operate a FC mechanism to prevent packet loss as outlined earlier. These links may be realized over a backplane as shown in Fig. 2.3 such that the SF is packaged on a single board or by cables as shown in Fig. 2.4 such that the SF is packaged across multiple racks. While a backplane is more efficient in terms of cost, space and in the round-trip time (RTT) bandwidth product and is therefore the preferred solution, cables offer higher bandwidth per link, which is very important for high-speed switching.

To underline the importance that future VLSI high-speed switching is constrained by 1) limited number of signal I/Os per chip 2) limited number of boards per backplane, we present an example of IBM's current switching family, the IBM PowerPRS [40] in Appendix B.

As a result cables as shown Fig. 2.4 will be employed. However, such "long distance" cables dominate the total SF costs. These cables are about 3-10m in length. Including the data integrity and synchronization processing of the link, RTT times bandwidth product between SA

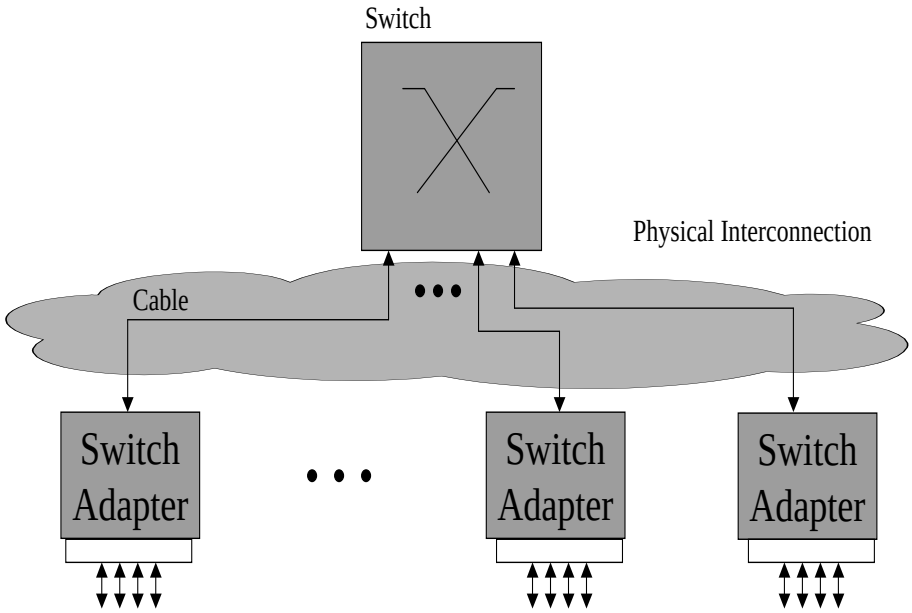


Figure 2.4: Next Generation of Switch Fabrics

and switch is expected to be in the 10s of packets. This is in contrast to what previous switch design work assumed. For instance [1] [37] [5] [6] [46] consider RTTs of 1, [47] in addition analyzes RTTs of 2. Large RTTs have so far solely been analyzed in the context of point-to-point links between SAs. Thus, while the **inter** SF communication for $RTT >> 1$ has been treated for instance by in [8] [10], the **intra** SF communication has been thoroughly neglected. In our thesis we will deal with this aspect and assume an RTT times bandwidth product of 64 packets, if not stated otherwise. For brevity, we say RTT=64. Our switch *design point* is specified work-conserving (definition provided in Subsection 4.1.1) for arbitrary input-output pairs using an RTT $= 64$.

In the following the functionality of the switch adapter and the switch are explained.

2.1.3 Switch Adapter (SA)

The SF interface, the SA, is another queueing point. It is a device that serves several purposes.

- **Queueing:** It performs input queueing for reasons explained earlier.
- **Segmentation/Reassembly:** It segments the frames of the NP into packets before and reassembles them again after switching.
- **Header Generation:** It compiles the routing and QoS information provided by the NP into a format that the switch understands and prepends it as packet header.

Sometimes the functions of the SA are a physical part of the NP, sometimes these functions are realized in separate functional SA units. A separate SA allows to interface to different NP vendors, standards and packet formats and gives hence more flexibility, but typically adds another queueing point. A SA incorporated into the NP gives therefore shorter packet delays through a networking node, but makes the NP even more complex and less flexible. As the combination of input and output queueing and the FC applied inbetween, is the essence of this thesis, we consider the SA as logical entity, which is part of the whole SF. The input queues are often realized as virtual output queueing (VOQ) , i.e. queuing at the input per output, to avoid

the HOL blocking phenomenon. Virtual output queueing is a well established way to implement input queueing. It has been adopted in the design of crossbar switches [89], as well as in the design of buffered switches using shared memory [44, 96] and will also be assumed in our work. This approach has been proven to be robust against varying traffic characteristics [38].

The segmentation/reassembly, as well as the header generation are the reason why the switch has to run faster than the targeted data line rate (speed-up). Note, that from this point of view the SA can also be considered a speed-matching entity. The SA may also convert the physical format of a packet, for instance from a serial to a parallel representation to efficiently interconnect to the switch. This depends on the link technology that is used. At the egress the SA again queues the packet to restore a format that is understood by the NP. The SA for one switch input/output port pair typically is one physical entity.

It is assumed that the adapter at the egress of the switch, the output adapter (OA), is always able to absorb all traffic at all times. This assumption is made in order to avoid interference with *congestion*. One purpose of our work is to find an optimal solution for resolving output contention, as we describe in Chap. 3. We therefore need to exclude occurrences of network congestion which is achieved by this assumption. Therefore, the OA is not relevant to our work. The adapter at the ingress side of the switch, the input adapter (IA), however, does play a major role in our considerations, since it houses the VOQs and the link (data path) scheduler. This is the reason, why we often see only the input aspect of a SA, and refer to it as IA. Therefore, we use the term IA and SA interchangeably. We exclude link technology issues from our work and assume an error-free physical interconnection between IA and switch. This is a reasonable assumption with current link technologies and coding schemes used. IBM's Unlink technology for instance achieves a bit error rate of up to 10^{-18}. For a 1Tb/s system, there would be one single error per roughly 11 days. With an additional Error Correction Code (ECC) for both the payload and the header the link can be considered error free. It makes therefore sense to assume responsibility of guaranteed delivery of packets by higher level protocols, e.g. end-to-end protocols that operate between NP-egress and NP-ingress.

2.1.4 Switch

The CIOQ class of switches is represented by two alternative architectures. One architecture is based on a cross-bar employing a centralized scheduler. The other architecture is an output-queued-switch based architecture. The former is attractive due to its simplicity and low cost in realizing the SF. However it is known not to scale well to larger fabric sizes. This is because it has to find a best match out of a set of N^2 inputs (VOQs) and N outputs (challenges for bipartite graph matching (BGM) algorithm). It is also, because there need to be $2N + N^2$ interconnection lines for the request and grant operation to and from the input adapters and the switch ports. Moreover, for achieving throughput of close to 100% with this architecture, a speed-up of 2 is required [79]. This speed-up is to be seen in addition to the speed-up necessary for the packet overhead. Therefore, the bandwidth through the routing fabric and the read bandwidth of the input buffers must be doubled. Considering the chip I/O bandwidth bottleneck, this speed-up, the wiring complexity, and the performance of the BGM algorithm are significant implementation challenges which exclude this architecture from our field of exploration.

The latter architecture is often realized with output queues located in a block of shared memory. This approach allows a better utilization of the limited memory space available on the VLSI chips, because the space allocated to a specific output queue varies dynamically with the load. However, mechanisms need to be put in place that avoid one output taking the maximum share thus degrading the performance of other connections. In addition the shared memory must also be protected from running out of storage. These FC mechanisms are costly in terms of packet

overhead and hence speed-up and bandwidth.

FC is the mechanism employed within SFs to fulfill the lossless requirement. It means the SF must not drop packets due to overflowing buffers. While some traffic classes of network protocols would allow the loss of packets, for instance, User Datagram Protocol (UDP), or Available Bit Rate (ABR) services in ATM, some of them don't, such as Transmission Control Protocol (TCP), or just allow the abortion of packets with a specific bit in the header being set, as for instance in Variable Bit Rate (VBR) class of traffic in ATM. From statement 2) - the switch to be network protocol independent and it to be able to support all classes of traffic - follows that the switch must not drop packets. This is not a necessary conclusion, since specific drop information might be passed from the NP level down to the switch. The implications of such a drop function don't directly relate to our work and are therefore shortly discussed in Appendix C. The output-queued-switch based CIOQ architectures has three drawbacks.

1. It requires multi-port queues that are challenging to implement.

2. For performance and economic reasons, these queues are realized within a shared memory, which requires a two layered FC mechanism. One to protect the output-queues and one to protect the memory from overrunning.

   Furthermore, a shared memory has physical implementation problems when increasing the number of memory locations, which is for instance necessary for larger RTTs. The problem is that the wiring to interconnect all input ports to all storage locations (input router) and the mirror-like construct on the output side (output router) is so immense that either novel technological or engineering approaches are required to overcome these problems. In [93] the problem is illustrated for a 32×32 switch device with 1024 memory locations, which would require 1 million wires for the input router alone making it impractical to implement.

3. In CIOQ architectures the order in which packets are read out of the queues is given by packet arrivals. Since multiple independent input queues are involved in possibly writing to one output queue, there is a high correlation in-between distinct flows. A much better technique would be to queue per input at the output. This gives the flexibility to install different output queue schedulers according to different traffic needs. We will analyze this more closely in this thesis.

An architecture that combines input and output queueing without having above mentioned drawbacks is the so-called combined input cross-point queued (CICQ) architecture, which is also known as buffered crossbar. The drawback of this architecture is that it requires N^2 queues within the switch, instead of N, as in the CIOQ case. Advances in CMOS technology do not hinder any longer feasibility of such architectures, as for instance demonstrated for a 64×64 SF in [41]. Therefore, we will choose a CICQ switch architecture as our switch architecture of choice.

We consider our analysis to be independent of service or traffic classes. With a CICQ architecture it is only a matter of data path scheduling techniques that are known from the NP level to implement different QoS classes at the switch output side. We assume the presence of one service class only.

Another property of the SF is its capability in delivering packets in order. This is a feature required by statement 1), for connection-oriented services such as real-time rt-VBR or Constant Bit Rate (CBR) of ATM and is realized by a FIFO service discipline of the individual queues. If packets could be dropped, in-order delivery would be much more difficult to guarantee.

2.1.5 Summary

To sum up, for a communication systems network, we assume a standard architecture as first described by [34] and [96] that comprises:

- a SF that operates based on packets
- VOQs at the SA (IA)
- the link between IA and switch is considered error-free
- single stage SF
- a switch that is loss-less and provides in order delivery of packets
- the NP to be always able to absorb the switched traffic

In this thesis we additionally assume:

- the switch to operate only on one single service class
- a buffered cross-bar switch memory architecture
- an RTT of several 10s of packets between SA and switch

Our switch design point is work-conserving for arbitrary input and output pairs using an $\mathrm{RTT} = 64$. We neither investigate in physical layer issues, nor in networking layer issues. Instead we will focus on the data link layer.

Protocols on this layer, but also on higher layers, that offer ordered, loss-free, and error-free delivery of packets ("guaranteed delivery"), also required for our SF, fall into the category of automatic repeat request (ARQ) protocols. ARQ protocols typically consist of these elements [112]:

- FC to provide losslessness
- loss detection, e.g. by time-out, or by gap detection
- re-transmission strategy, e.g. selective repeat, Go Back N, etc.
- acknowledgements, such as cumulative, positive or negative

As we assume an almost error-free link we relegate the problem of guaranteed delivery (last two bullets) to ECCs and higher layer protocols and will focus on the FC aspect (first bullet). Nevertheless, our system is expected to be able to detect errors, which is important for fault isolation (second bullet).

2.2 Multiprocessor Environment Example

For a multiprocessor system the same assumptions apply as in a communication system. Additionally, the computing nodes and the targeted applications must be more concretely specified, since they are influential on communication patterns and hence system performance.

A multiprocessor (MP) environment is a system of computing nodes that are interconnected by a network topology. For interconnecting a small number of processing nodes, e.g. $<= 8$, a bus topology is sufficient. For medium sized networks, i.e. 8 to 512 processing nodes,

busses show severe performance degradation [109]. Therefore, ring networks (NUMA-Q [64], NUMAchine [92], KSR-1 [68]) or switches (SP-2 [81], Spider [88]) are preferred. The former may be used as a single ring or a hierarchy of rings. The latter may be used as a single device, or multiple of them may be interconnected to – for example – form a fat-tree (SP-2), or a Hypercube topology (Spider). Interestingly, the multiprocessor environment looks very similar to the communication systems environment as depicted in Fig. 2.5, where the queueing points of a MP environment are shown. The Network Interface Adapters (NIA) in this figure incorporate the SAs. Physical separate SAs are avoided, because extra queueing points increase node-to-node delay, which is unwanted. It is important to note that the interconnection between NIA and switch is also done by cables that may be up to 10m long, as is the case for the next generation communication switches. For proof of concept in this thesis, we will operate such systems using a single switch, and limit the applications to run on at most 32 processing nodes in parallel. We want to note that such systems fall into the category of non-uniform memory architectures (NUMA) that have a global memory that is shared, yet evenly, physically distributed among all nodes. Therefore, each node has two possibilities to access data from the memory: local accesses, which occur transparent to the other nodes, and remote accesses that do require network interaction.

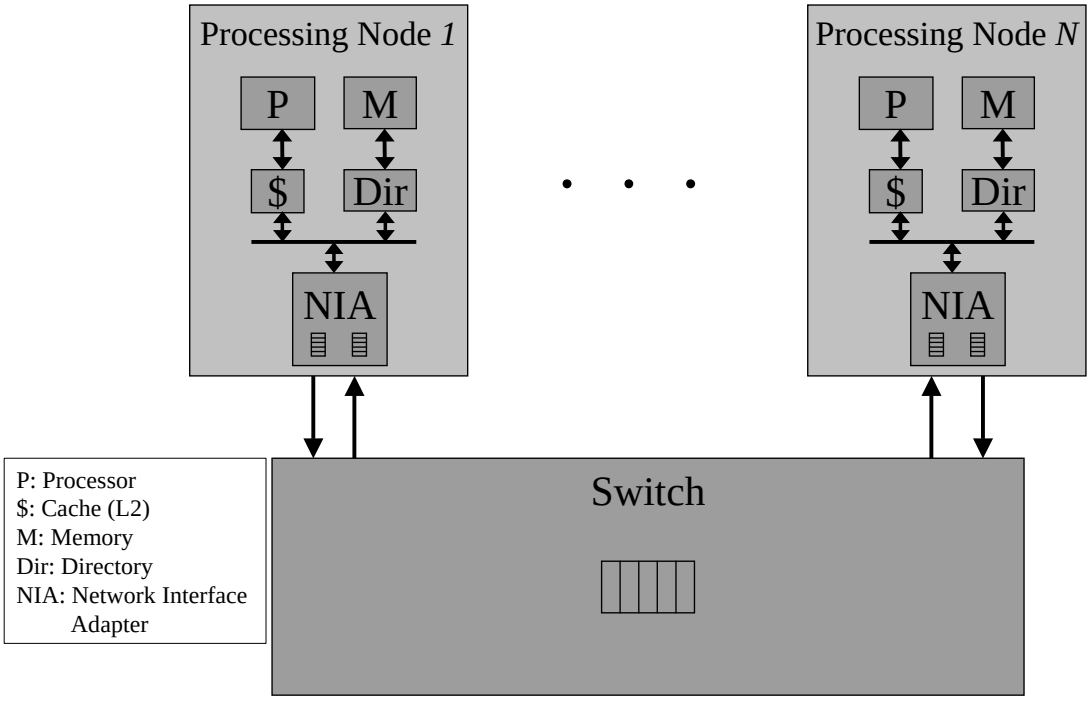


Figure 2.5: Multiprocessor Network

The NIAis responsible to set up the request, reply, and control messages as directed by a directory-based cache-coherency protocol, which is common in NUMA architectures and is assumed in our case. For an overview of cache coherency schemes we refer the reader to [107], and [108]. Specifically, we run a full-mapped invalidation-based directory scheme and apply a MESI [99] (Modify Exclusive Shared Invalidate) cache-coherency protocol to operate between the nodes. Control messages comprise acknowledgements, as well as coherency, or replacement messages. A NUMA belongs to the class of Shared Memory Processing (SMP) computers, which is known to generate more interconnection network traffic, than for instance distributed (shared) memory computing that uses message passing to share data [103]. This is what we aim for, because we want to stress the capacity of our interconnection network, i.e. the SF. Since

we do not perform any classification for different QoS purposes to the messages of the MESI protocol, the MESI protocol is not a limitation to our findings. From a protocol perspective it means that an end-to-end protocol is running on top of the network at its terminal points (see also Fig. 2.6). We consider the traffic arriving at the NIA as given. We acknowledge that the traffic pattern, specifically the message interarrival time, the message size distribution, and the message destination distribution depends on system parameters such as cache size, memory size, memory layout, cycles per instruction of the processor, cache-coherency protocol, etc. We do not want to study the impact of such parameters. We therefore assume a fixed set of system parameters. This set has been selected according to [71], and represents parameters that produce good overall performance. Our assumptions of the system are listed in Table 2.1.

Table 2.1: Multiprocessor System Parameters

Module	Parameter	Description
Processor	Clock	300 MHz
	Number	4, 16, 32
Cache	Size	1MB
	Set	4-way associative
	Line size	64B
Bus	Memory access	3 processor cycles
Memory	Policy	first-touch policy to allocate data at a cache-line granularity to nodes
	Pages	no virtual memory
Messages	Request	16 bytes (= 1 switch packet)
	Reply	80 bytes (= 5 switch packets)
	Control	2 bytes (= 1 switch packet)

Another influential factor on system performance is *workload*. Depending on the application and its problem size, traffic patterns will change. Therefore, we choose with the SPLASH-2 application and kernel suite [70],[71] a representative benchmark. We choose the applications, and problem sizes according to Table 2.2.

To sum up, for a multiprocessor environment, we additionally assume:

- a NUMAmultiprocessor environment.
- the operation of the simplest switch topology, which is a single switch.
- fixed system parameters and use 5 applications out of the SPLASH benchmark as representative workload.

We exclude topological and as a consequence also routing issues. We want to underline that we do not want to analyze the impact of system parameters of the processing nodes. Therefore, in this thesis we will analyze the switch's behavior exposed to such traffic.

2.3 Unified View

By just looking at the interconnection aspect of both the communication and the multiprocessor system, as outlined in the previous Sections 2.1 and 2.2, we can derive a unified view of the

communication subsystem. A NP in the communication world corresponds to a processing node in the multiprocessor world. Both are terminating packets that have been set up for the purpose of switching. From the SF point of view these terminating nodes are the source of traffic. We will define the specific traffic sources later in Chap. 2.5. While the SA in communication systems is most likely a separate device in that sense that it is physically separated from the NP, in multiprocessor systems the NIA bears the same functionality and is embedded on the processing node. Both systems use long cables to interconnect the SAs to the switch.

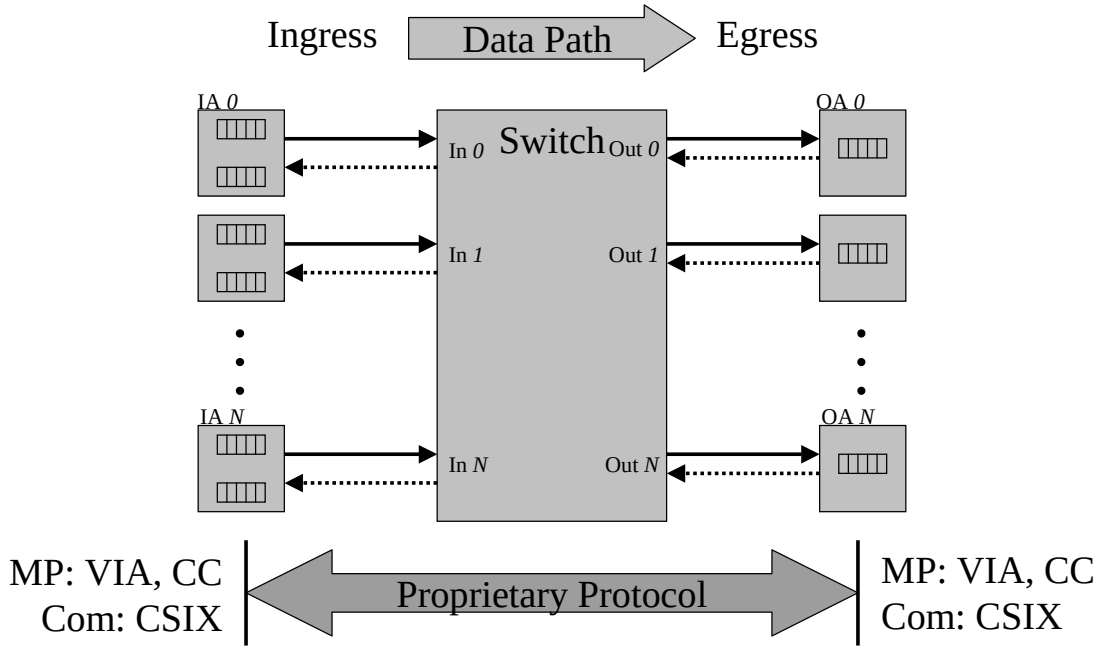


Figure 2.6: Unfolded Switch Fabric

The difference between the two systems is therefore determined by protocols that run "outside" of the SF, i.e. its adapters. The commonalty between the two systems starts and ends at these adapters. This is also shown in Fig. 2.6, where the SF is unfolded and the adapters are split into their input (IA) and output part (OA). We see, as an example, that for communication systems (Com) the CSIX interface would reach up to the adapters, while for MPsystems it would be a MESI, Virtual Interface Architecture (VIA) [26], or InfiniBand protocol for instance. We view the space in-between as our area of interest, which is indicated in the Fig. 2.6 by "proprietary protocol". The unfolded view of the switching system allows to display the flow of data from left to right, i.e. from ingress to egress. The *control path* therefore goes from right to left, as indicated by the dotted lines. The control path is necessary to transport the FC information. FC is mandatory, as we assume a loss-less switch as already pointed out in Sec. 2.1. A further abstraction of this model can be made, when taking the earlier made assumption into account that the NP is always able to absorb all switched traffic. This assumption makes the FC channel between OA and switch superfluous, which is also reflected in the resulting model, shown in Fig. 2.7.

We will from now on use this model for our simulations. The model shows further the input traffic sources S_i and assumes a single switch of dimension $N \times N$, whereby N signifies the number of input and output ports respectively.

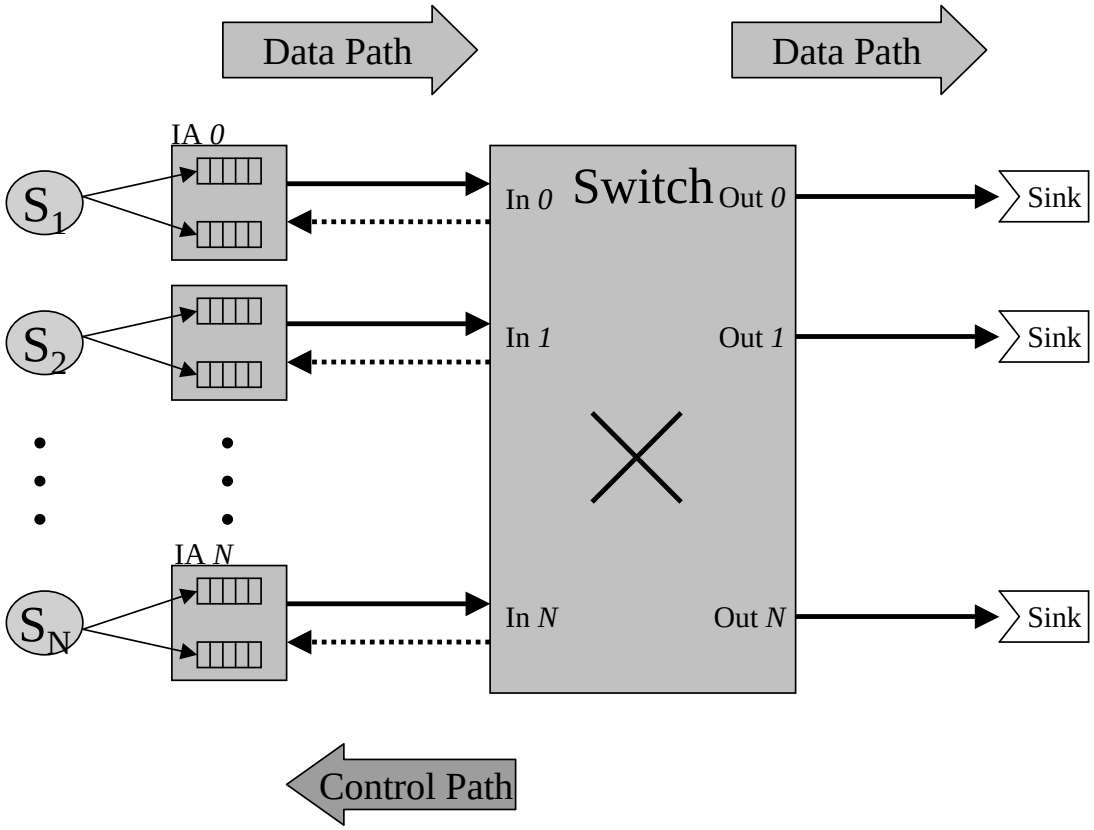


Figure 2.7: Abstracted Switch Model

2.4 Simulation

After having presented a unified model of the two targeted systems, we give an overview of the simulator, the simulation techniques, and the applied statistical means. The latter two are the most crucial aspects of simulations according to [74]. We will show that our simulations meet the standards set by this article.

We use Hyperformix's SES/Workbench to do our modelling. Workbench is a discrete-event simulator suited to model queueing systems. It provides a predefined set of elements, called nodes, such as queueing elements with configurable service time or discipline to temporarily buffer messages, which themselves are represented by transactions. SES/Workbench allows to allocate resources to model the utilization of memory locations, for instance. It further allows to structure the model into subentities, called submodels, that are used in our case for the representation of the IAs or the switch. Certain functional islands within these submodels, such as the output queue scheduler, or the control path (=reception) scheduler are compounded in a node that allows to specify algorithms in C. SES/Workbench supports sequential simulation as well as simulation by independent replication. Each variable that is statistically enabled delivers all the momenta, including minimum and maximum value as well as the number of samples. With any replicating simulation techniques the 90, 95, and 98% confidence intervals with the respective errors are given.

We use the default pseudo random number generator, which is linear congruential with modulo $2^{32} - 1$. It is implemented according to [102] and used in our model to draw numbers for the packet generation process (for communication systems), which is based on a Markov-chain with two states (`ON` and `OFF`) to simulate bursts as described in Chap. 2.5. There are N independent traffic generators S_i, one for each port, as sketched in Fig. 2.7. Therefore, per unit time

each traffic generator draws one random number to determine its next state. In case of a state change from OFF to ON another one is drawn to determine the packet destination. We need to make sure that the random numbers are not wrapped by the pseudo random number generator during long simulation runs. In the worst case of 100% offered load, there are $2N$ random numbers generated per unit time yielding $X = N$ new packets. Consequently, there are

$$X_{\max} = \frac{2^{32}}{2N} N = 2^{31} \tag{2.1}$$

packets allowed per simulation run without endangering the results, which is obeyed in our simulations.

In order to decide which simulation technique to choose, we have to define the purpose of our simulations and the main metric that we want to capture.

The purpose of these simulations is to evaluate several control path scheduling strategies and their impact on performance under varying system and traffic conditions. To this end, we run the simulations for a given number of packets and measure the average system delay from packet entry at the IAs to their exit at the switch's output ports. The system delay that we measure is not a delay specification meaningful for later operation. It is rather our metric of evaluation. For those algorithms that have a lower delay than others we readily conclude that they are better. After the simulations, we therefore are able to state which strategy is superior over another.

In order to see the difference between these schedulers, we have to operate the switch at saturation, $L \approx \rho_{\max}$, with L denoting the offered load and $\rho_{\max}$ the maximum throughput. If we run above saturation, then too many packets will be buffered in the VOQs, which eventually leads to exhausting simulation resources. If we run too much below saturation the schedulers are not challenged. We have found that - although depending on the burst size that is applied - a reasonable assumption for the switch is to have a throughput of around $\rho_{\max} = 0.98$. If not stated otherwise, we exercise the switch for this reason at an offered load of 98%, $L = 98\%$. This is not a limitation in the interpretation of the results, because what is observed at a load of 98% will also be seen at higher loads, i.e. 99, or 99.99%. At a load of 98% we will be able to sufficiently well demonstrate the scheduler behaviors, yet run the simulations at a steady state.

The scheduler will specifically be challenged by bursts. In order to obtain statistical relevance, we want to make sure that the bursts simulated per cross-point are in the hundreds. This has implications on the total number of packets simulated for varying switch sizes. We are interested in switch sizes of $N = 8$, 16, and 32. Assume a given number of packets n that are on average – independently of the selected switch architecture – simulated per cross-point. Then the number of packets simulated per port $X_{\text{port}} = nN$, or equivalently for the whole switch $X = \sum_{i=1}^{N} X_{\text{port}} = nN^2$. With $n = 16,000$, we have over 500 bursts for a burst size of $BS = 30$ (on bursts and burst sizes see Sec. 2.5) independent of the switch size. This meets our target and will deliver statistical relevant data.

The simulation techniques that we use for the collection of results of synthetic traffic is the one of independent replication. There are multiple simulation runs. Each simulation run consists of three phases, the warm-up, which is needed to populate the model and the buffers, the running phase, during which statistical data is collected, and the ending phase, where the switch is being emptied and it is checked whether all packets were switched correctly through the device. The end of the running phase is found, once all X packets have been generated. A warm-up phase of 1'600 packets per port have shown to bring the switch close to $\rho_{\max}$. The total number of packets X are counted from the beginning of the running phase.

In order to reach confidence on the results, we aim at a half-sided confidence interval of 3% of the system delay at a confidence level of 90%. With other words, simulations are stopped as

soon as the relative error on system delay collected over multiple runs falls below 3%.

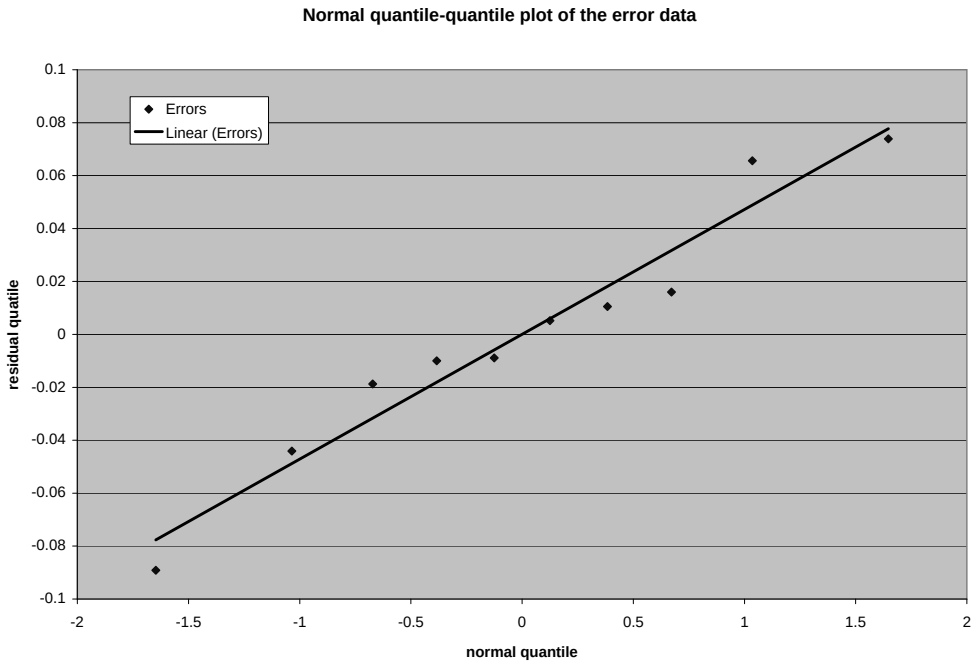


Figure 2.8: Normal Quantile-quantile Plot for the Error Data

The confidence intervals are determined using a student-t distribution, if the number of samples (independent runs) is less or equal than 30. The student-t distribution is a valid assumption if the errors are uniformly distributed (see [97]). We therefore validate this requirement graphically by using a quantile-quantile plot and taking exemplary one data point from our simulations conducted. The system parameters for this data points were a cross-point memory size of $M = 13$, $\mathrm{RTT} = 64$, Load $L = 98\%$, control path scheduler (=reception scheduler) scheme = FIFO, and a burst size B=10. Confidence was reached after 10 replications. The results are shown in Fig. 2.8 and demonstrate that the errors are indeed uniformly distributed.

2.5 Traffic

In this section we describe the traffic sources that were applied. We use two different types of simulations:

- synthetic traffic
- trace-driven simulation

While the former one is used to model communication system traffic, the latter is used as multiprocessor workload.

Why not execution-driven simulation?

Researchers that work on the improvement of the execution time of parallel application frequently use execution-driven simulations. These type of simulations are the most accurate, as they capture the real interaction between processing nodes. Typically, the network is abstracted by a model, i.e. the LogP model [75], [76] and the focus is on the influence of system parameters. However, our focus is the network, which we need to model in detail. Therefore, for

our purposes we had to model such a system in full detail from the instruction set down to the network protocol. Initial tries of this approach showed that there are severe memory implications to the simulator, allowing only systems of up to 8 nodes to be modelled with extreme long simulation times. We refrained from this approach, because we want to model not only 8, but also 16, and 32 node systems. For this reason, we can not use execution time of an application as a metric to compare the quality of our FC optimization effort. Instead, we use the average cell delay and link utilization as our metrics (see also Subsection 7.6.5).

2.5.1 Communication Traffic

Traffic for communication systems is abstracted by the following three properties, each of which typically modelled by independent random processes.

- temporal variations
- locality
- communication volume

Temporal variation is expressed by the message generation rate. We use three models for its determination. One is referred to as Bernoulli, the other one is referred to as bursty, and the third one is referred to as bimodal traffic. All three are explained in the following.

Locality is specified by the distribution of message destinations. Destinations are determined according to a uniform distribution. A non-uniform distribution of packet destinations is also being presented in this subsection.

Communication volume is determined by the number of messages generated and the distribution of message length. We assume fixed size packets throughout the simulations. The number of messages generated is determined by the condition to terminate a simulation run (see Sec. 2.4). We simulate typically for burst sizes of $BS = 10, 30$. These burst sizes are justified by the two dominant modes (565 bytes and 1500 bytes) observed in the internet[78] and mapping these into the switch packet payload . The expected packet size that the switch is capable in switching is up to 64 bytes. This results roughly in burst sizes of $BS = 10$ for the mode of 565 bytes and $BS = 30$ for the 1500 bytes mode. For the purpose of interpreting trends in our results, we also simulate for burst sizes of 60, which might be justified by jumbo-frames in IP.

Bernoulli Traffic

In Bernoulli traffic the sending of packets is determined by a Bernoulli random process, whereby the probability of success (sending a packet) p equals the load L, hence $p = L$. A Bernoulli experiment is conducted per input per unit time. Each packet draws a new destination from the destination giving process. Hence, there is no time-destination correlation and the arrival rate of subsequent packets is exponentially distributed.

Bursty Traffic

The bursty traffic model is characterized by a two-state Markov chain process that is able to determine the overall load L, and the burst length BS. This traffic pattern exhibits a time-destination correlation, called bursts.

The Markov chain consists of an `ON` and an `OFF` state as shown in Fig. 2.9.

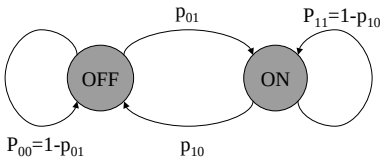


Figure 2.9: Two-state Markov chain

During the ON state, S_{ON}, traffic is generated at line rate $\lambda_{\mathrm{ON}} = 1$, i.e. packets are sent back to back. During the OFF state, S_{OFF}, no packets are generated at all, i.e. $\lambda_{\mathrm{OFF}} = 0$. The line is idle during this time. After each transition from OFF to ON state described by probability p_{01} a new destination is determined according to a uniform distribution. The destination remains unchanged for all packets that are generated during this state. Therefore, the average time spent in ON state, $\overline{S_{\mathrm{ON}}}$, is equivalent to the average burst length BS. The state transition probabilities as a function of burst size BS and load L are calculated in Appendix D

Bimodal Traffic

Bimodal traffic is not only observed in the internet, for IP traffic, but also in system area networks (SANs), e.g. FibreChannel networks and, NUMA environments [72]. The two modes are distinguished in terms of frame length, whereas a frame consists of multiple packets. Having a synthetic traffic model, accounting for the distribution of frame length, combined with an appropriate distribution of destinations and a valid packet generation rate per input, all possible traffic sources could be modelled. The generation of frame length as described in [72] is one step towards this goal. By applying a uniform destination distribution and a bursty temporal variation, a good approximation for IP traffic can be made. However, applying this traffic source to our unified switching model did not reveal any new insight, even if breaking down the system delays achieved into the two modes of packet lengths. The results were comparable to a modestly bursty traffic source

Non-Uniform Traffic

We introduce a traffic pattern that models non-uniform locality. We follow hereby the approach of [49]. In multiprocessor environments for instance it occurs frequently that one port is favored over others. Such a behavior is for instance noticed in the FFT kernel [77]. But also in communication environments, for instance in LANs, it could occur that a large file is or a multiplicity of files are being transmitted for example during a backup, thereby favoring the destination port that hooks to the backup server. In this time traffic to other destinations is low. Hence, there is an unbalance between traffic that goes to the favored and traffic that uniformly proceeds to all ports.

Mathematically speaking, there is an input load L that is distributed with a probability w, the so-called unbalanced probability, to the favored port, while the remaining traffic $1 - w$ is uniformly distributed over all ports N. Therefore, the traffic load from one input port i to output port o, $L_{i,o}$ is given by

$$L_{i,o} = \begin{cases} L(w + \frac{1-w}{N}) & \text{if } i = o \\ L\frac{1-w}{N} & \text{otherwise.} \end{cases} \tag{2.2}$$

The aggregate offered load that goes to output o from all input ports, L_o is given by

$$L_o = \sum_i L_{i,o} = L(w + N\frac{1-w}{N}) = L \tag{2.3}$$

Eq. 2.3 shows that output ports are not oversubscribed. When varying the unbalanced probability w, we obtain uniformly distributed traffic for $w = 0$ and totally directed traffic for $w = 1$, because here only traffic that satisfies $i = o$ is generated.

2.5.2 Multiprocessor Workload

Multiprocessor workload is being modelled using traces. The traces have been collected using the RSIM simulator [73]. The system parameters have already been specified and are listed in Tab. 2.1.

A comprehensive list about the advantages and disadvantages of simulation by traces is found in [97]. From this list we find that traces have the advantage of a fair comparison between two alternatives as they have a deterministic input. As all other parts of our model are not random as well, we acquire absolute results. This is the reason why we do not have to compute a statistical confidence on those results. They are simply not necessary. One has to be careful with traces, because they are strongly dependent on system parameters and results may become misleading. Also as workload changes over time, they may become obsolete rather quickly. These arguments do not apply in our case, as we do not want to demonstrate an improved application performance by changing some SF parameters. We therefore do not model an interaction of trace events and the system under test. We use traces solely as a generator of traffic, with a certain temporal and its specific spatial behavior. Due to this exclusion of interaction of trace events and the system under test, we are also able to modify the temporal behavior of the trace by obeying its order of events. We can run the trace faster ("squeeze the trace") or slower ("expand the trace") in order to put more pressure on the SF or to relax it. We refer to this parameter as *trace speed-up*. Typically, the former is more interesting. By doing so, we keep the relative message generation time conserved.

Since one trace is only a single point of validation, we use five shared memory applications from the SPLASH/SPLASH-2 suite of benchmarks: FFT, Water, Radix, LU, and MP3D with the specifics listed in Tab. 2.2 to validate the results.

These applications and their expected communication characteristics are described in Appendix E.

Table 2.2: Application Parameters

Application	**Parameter**	**Problem Size**
Radix	Integers	52488
	Keys	524288
FFT	Points	64k
LU	Matrix	256×256
	Blocks	8×8
MP3D	Particles	50000
Water	Molecules	512

Chapter 3

Objective

In Chap. 2, we derived a model of a generic SF that we want to use and which is applicable in both communication and multiprocessor environments. This SF consists of N IAs and one switch. We also characterized traffic in both environments, demonstrated how it is generated, and presented the simulation technique. We found that traffic is unpredictable in terms of temporal and spatial variations. The only prediction that can be made is that contention due to these variations will occur.

It is the objective of the switch to resolve such contention phases quickly as well as affecting neither fairness - as a qualitative statement - nor throughput - as a quantitative statement. Contention significantly contributes to system latency.

As switches grow in number of ports and in aggregate throughput over time, switch cost is increasingly dominated by interconnection technology as outlined in Section 1.2 and Subsection 2.1.2. Interconnection technology, i.e. the amount of bandwidth that can be moved per square inch across chip pins and per inch across card edges, becomes the main bottleneck. At the same time more transistors per chip area are available, as density goes up. Therefore silicon becomes cheaper relatively to the interconnection technology. As a consequence, the interconnection technology becomes the dominating cost factor in switch design [95][93], resulting in the urge to use bandwidth more efficiently.

Bandwidth utilization is also determined by the packet overhead, which mainly comprises FC information. FC information in turn depends to a large degree on the system size N, as it scales with $O(N^2)$.

> Therefore, one objective of this thesis is to optimize the FC overhead to achieve better bandwidth utilization and to optimize output contention resolution.

We strive for a solution that with some extra on-chip intelligence allows the FC overhead to be minimized and to intelligently resolve output contention situations. Therefore more complexity will be the trade-off for less FC bandwidth. Higher data rates imply larger RTTs between switch and its associated adapters. Therefore, FC information is no longer available instantaneously. This time lag between the switch entities typically requires larger buffers and has an unknown impact on performance.

> Therefore, another objective of this thesis is to comprehensively quantify FC complexity, information and resource consumption for various FC schemes, and study the implication of large RTTs.

CIOQ switches have been the state of the art in switching for some time. The CIOQ alternative that uses a crossbar suffers from the control overhead due to its centralized arbiter. In addition, an iterative matching algorithm such as iSLIP [91] takes more than three clock cycles to converge. Its scheduling design rule is deterministic: In every packet cycle a best match between requesting input and granting outputs is to be found. This requires that the complete (buffer) state information of the switch be available at the input schedulers and considered. The CIOQ alternative using an output-queued switch suffers from memory-access capacity limitations. In addition, its FC overhead is quite large. The scheduling design rule is non-deterministic: Packets are always accepted into the switch until the buffers are full and FC is activated. This basically requires no (buffer) state knowledge at the N input schedulers, except eligible and feasibility signals. The buffered CIOQ alternative is more expensive to realize, because of the on-chip memory, but shows the best performance. Lower cost is therefore the main advantage of a bufferless crossbar.

A third objective of this thesis is to find a compromise between those extremes for the SF:

full state information	low cost	moderate performance
no state information	high cost	good performance

We will show that we can achieve a good compromise by using a distributed scheduling architecture that has the performance advantages of a buffered CIOQ, entails moderate cost, and utilizes partial state information.

So far, the switch has mainly been understood as a device that needs to be optimized for aggregate performance. A global output queue per output port was therefore appropriate for buffered switches. With the changing needs of the Internet, in particular to handle an increasingly demand for QoS aspects in the network, the switch is also exposed to the effects these changes entail. Also, in multiprocessor systems there is a need to distinguish flows by the type of message they carry. The Internet is characterized by a non-cooperative environment with many flows competing for link bandwidth, whereas multiprocessor systems represent a cooperative environment, because multiple processing nodes work together on the same problem.

A last objective of this thesis is to provide insight into the fundamental requirements of output-queue scheduling.

We will approach the overall problem of reducing FC overhead and maintaining SF performance by subdividing it into smaller problems and solving each separately, and then reassembling the solutions. As each switch has an input-serving and an output-serving process, we will subdivide our problem into a problem of the input side and one of the output side.

As we have discussed in the last chapter, the presence of contention has to be considered explicitly, in order for FC to be activated. Contention occurs when several inputs send traffic to the same output within a packet interval. In the light of different environmental needs, we will analyze the effects of two different, but well-known scheduling disciplines: round-robin (RR) , and oldest cell first (OCF) .

Because the output side is not being able to serve at the input rate of up to N input flows, resources become scarce. Once resources are scarce or even fully depleted, the input side is no longer able to send packets. This is when FC will be activated. The purpose of FC is to perform resource management [33]. One aspect of FC is therefore to provide a loss-less SF, which is its traditional purpose. The second aspect, which is the major contribution of this thesis, is that

proper resource management allows one to maintain and – under certain traffic situations – even improve overall switch performance. This analyzes the input side of a switch.

The remainder of this thesis is organized as follows. In Chapter 4, we will review existing FC schemes under the aspects of resource consumption, FC information and complexity. Next, we will subdivide the switch FC entity into smaller entities on which we can work separately in Chapter 5. The results allow us to summarize the FC overhead for different switch architectures, already considering our main contribution. We are even able to formulate a new subclass of CIOQ switch architectures that was discovered by our results. We do this in Chapter 6. Finally, in Chapter 7, we will prove our contributions by means of simulations and demonstrate how we achieve the objectives mentioned here. We conclude our work in Chapter 8.

Chapter 4

The Flow-Control Design Problem

4.1 Basic Flow-Control

How to interconnect two or more buffers in a packet-switched network such that no buffer overflow occurs is a FC design problem, which we shortly refer to as *the FC problem*. This requires a feed-back loop from the receiver (current node) to the transmitter (up-stream node) to control the packet transmission process, i.e. to determine when to start or stop sending packets. This problem is shown in Fig. 4.1 in an arrangement of nodes called from the data path perspective up-stream, current, and downstream node. The current node is the node under investigation. The line interconnecting two buffers is the data path, or the forward path, while the feed-back loop is also referred to as the control or reverse path. Packets are transmitted over the forward path, while FC information is found on the reverse path. The sending of FC information is synchronized with the data path, i.e. the arrival of packets. This means that FC information is sent or updated in packet-cycle time units (tu). The maximum rate at which the link can transmit packets is $\lambda_{\max}$. If the link delay for a packet or a unit of control information I, respectively, is $d = d_D = d_C$, then the RTT expressed in packets is $RTT = 2d\lambda_{\max}$. It therefore takes RTT/2 packet cycles for a packet to propagate on the data path, and RTT/2 cycles for the control information to return. For the buffers, send and reception buffer B_s and B_r, we assume a FIFO service discipline.

After this general description of the basic FC scheme, we will proceed with some specific definitions in Sec. 4.1.1, and performance-related aspects in Sec. 4.1.2, such as the relationship between work-conservation and performance.

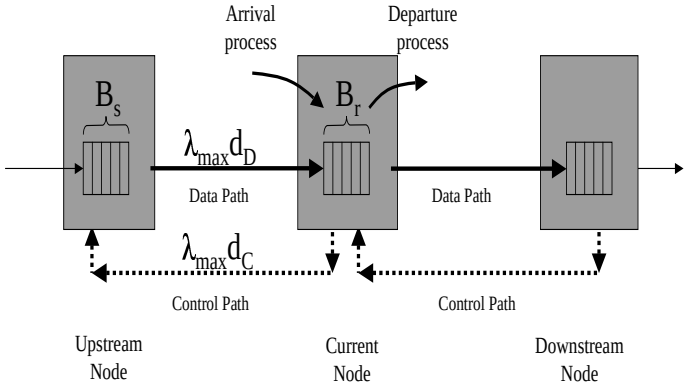


Figure 4.1: The General FC Problem

4.1.1 Definitions

There are two FC loops in the system of Fig. 4.1. The current node generates a control signal to be read by the up-stream node (*up-stream FC loop*) and it terminates the control signal from the downstream node (*downstream FC loop*). If packets are queued at a node, they are said to be available. A packet currently at the head of the queue is said to be *eligible*. For a packet to be dispatched, it must have the permission from the downstream node. The *feasibility* information carries this permission. We accordingly say the packet must also be *feasible*. Therefore,

> a packet is dispatched, if it is *eligible* and *feasible*.

> *Feasibility* is the permission from the downstream node to dispatch a packet.

We similarly say that in the presence of feasibility information the link is up or the node is unblocked, otherwise it is down or the node is blocked. Feasibility information is generated at packet cycle instances, and is caused by *FC events*.

> FC events are buffer state changes that are caused by packet arrivals or departures.

The process of controlling a node's buffer state to avoid overflow is active buffer management. A minimum buffer size $B_{\min}$ is necessary to provide correctness, i.e. to fulfill the lossless requirement of Subsection 2.1.4.

> An FC scheme is said to operate *correctly* if it prevents buffer overflow.

In order to be able to evaluate the efficiency of buffer management, we introduce the term *work-conservation*.

> A current node is said to be *work-conserving* if it forwards packets successively (back-to-back) at indefinite availability of packets at the up-stream node during feasibility periods assuming equal-capacity upstream and downstream links.

4.1.2 Performance

The node under investigation is the current node as depicted in Fig. 4.1. If a node is work-conserving then its efficiency is said to be one, $\epsilon = 1$. Efficiency is reduced, if a gap is observed at the output side of the current node. It means that packets are not available for certain time periods. As we assume indefinite availability of packets at the upstream node, and the downstream link to be up, the reasons for unavailability of packets at the current node are solely to be found in its generation of feasibility information for the upstream node.

Given a packet arrival rate $\lambda_{\max} = 1$ during feasibility periods, a time delay d can be translated into buffer space B using Little's law: $B = \lambda_{\max} d = d$[packets]. This allows us to deduce the number of required buffer locations from the time delay of feasibility information.

Performance Degradation

In the following we list reasons that lead to unavailability of packets at the current node and hence to performance degradation. Specifically, we give several reasons that explain the delay of feasibility information:

a) (For the sake of completeness) There is no traffic available even at the up-stream node. Note that this case is excluded by our definition of work-conservation.

b) We are still in a warm-up phase. The current node has not yet seen any packet. It takes "propagation delay" time until the first packet arrives, under the assumption that feasibility information is available up-stream at initialization.

c) The feasibility information is delayed. There are five factors that can contribute to this delay:

   α) *Restart Time* t_r: This time is relevant if the downstream link was blocked but now is re-enabled. Restart time is the time between arrival of feasibility information at the current node (downstream FC loop) and the generation of feasibility information for the upstream node (upstream FC loop). This time expresses how fast the current node reacts on the re-enabling of downstream feasibility information and can be considered as the link between the downstream and the up-stream FC loop. Provided a certain amount of buffer B_e, this time penalty can be hidden. For correctness reasons, B_e must be as large as to accept RTT packets, meaning that from a FC design perspective $t_r = \mathrm{RTT}$, even though during operation $t_r \leq \mathrm{RTT}$. Therefore, t_r is a time penalty paid for some systems to assure correctness. In Subsections 4.3.1 and 4.3.2 we give application examples. The following times relate only to the upstream FC loop.

   β) *Propagation Time* t_p: The time it takes for the feasibility information to progress between the two neighboring nodes, e.g. the current node and the up-stream node. This is RTT/2, and invariant. The propagation time is seen twice, as feasibility information has to return to the upstream node and packets have to arrive. Provided a certain amount of buffer $B_{t_p} = 2t_p$, this time penalty can be hidden.

   The propagation time has an impact on the timeliness of information. The longer it is, the higher the chance that the current state of the sending node deviates from the state of the receiving node. We will study the impact of outdated information in a later chapter.

   γ) *Queueing Time* t_q: The feasibility information generated is queued because it can not be drained fast enough. This could have the effect that FC events will queue. Queueing time relates to the fact that an FC channel may not provide sufficient bandwidth to convey all control information generated per packet cycle. The available FC bandwidth determines the rate μ_{FC} at which a certain amount m of FC information I is drained to the up-stream node per packet cycle. Hence $\mu_{\mathrm{FC}} = mI/tu$. A certain amount n of FC information I is generated per packet cycle tu, such that its generation rate is defined as $\lambda_{\mathrm{FC}} = nI/tu$. For the case that $m \geq n$, $t_q = 0$, because all information generated is drained instantaneously. For the case of $m < n$, $t_q > 0$. In this case, we describe the FC intensity as $\rho_{\mathrm{FC}} = \mu_{\mathrm{FC}}/\lambda_{\mathrm{FC}} = m/n$. Conservation of work is influenced such that the arrival rate of packets at the current node r adapts to the generation of feasibility information, i.e. the FC intensity. Hence, an initial rate of $\lambda = 1$ converges after one RTT to $\lambda = \rho_{\mathrm{FC}}$. This means that even though the

data links are of equal capacity, a bandwidth-constrained FC path may influence the performance of a system.

Whereas the FC link bandwidth provided is an engineering choice driven by technological constrains and cost, the amount of flow control information I depends on the FC scheme. As a system is typically designed for $n = m$, the amount of FC information I determines the FC link bandwidth provided and hence the link cost. We will analyze the FC overhead for basic FC in Subsection 4.2.1 and the relationship between link cost and FC overhead for FC in switches in Section 5.1. We can conclude that a smaller I is better, and assume in the following $n = m$.

δ) *Processing Time* t_s: This time is observed, when the processing at either of the two nodes, the current node or the upstream node, is larger than one. Provided a certain amount of buffer space B_s, this time penalty can be hidden. We assume in the following that $t_s = 0$.

ϵ) *Feasibility* inter-departure time t_n: This time describes the fact that feasibility information may not be returned in every packet cycle instance, thus producing gaps in the FC path. Assuming that FC information is sent at every nth instance, n feasibility information entities may queue up. Introducing this time delay has the potential to reduce the overall FC bandwidth by a factor of n. It is applied in those FC schemes in which n FC events may be combined into one, such that the FC information sent represents all n state changes. Provided a certain amount of buffer space B_n, this time penalty can be hidden.

The total delay $t_{\rm tot}$ of feasibility information is composed of the orthogonal factors $t_{\rm tot} = t_r + 2t_p + t_q + t_s + t_n$. In order to design practical systems, we have to translate these times into buffer space using Little's Law as mentioned earlier and $\lambda = \lambda_{\rm max} = 1$. With typically $t_s = t_n = 0$ and under the general condition of $t_q = 0$ for systems where $m \geq n$, we find the total amount of buffer space B_{tot} for the system to be work-conserving and correct to be $B_{tot} = B_e + B_{t_p} = 2{\rm RTT}$. A system that mandates correctness only, a buffer space of $B_{\rm min} = B_e + 1$ is sufficient. The one is necessary to make the system operational.

Test for Work-conservation

In the following we present a procedure that allows us to determine whether a node is work-conserving.

Let us consider a trivial period of RTT cycles that is counted from the first occurrence of feasibility information after a blocking period that lasted for at least RTT cycles. If the number of feasibility information instances is smaller than RTT then $\epsilon < 1$. The gap can be measured and is expressed as the *correction time*, $t_c > 0$. It means that operation was not work-conserving. However, the time t_c was introduced by the FC mechanism to assure correctness. Therefore, the condition that must be satisfied for FC to be work-conserving is written as $t_c = 0$. For an observed correction time $t_c > 0$, a buffer worth t_c packet locations must be added to achieve work-conservation.

4.2 Characterization of FC Schemes

Among the various FC schemes to choose from, we prefer the one that requires the least amount of implementation resources, because resources are expensive. The following resources relate

to the implementation of FC schemes:

- control path, i.e. amount of FC information,
- buffer size, and
- chip area

In the following we will briefly introduce and discuss the importance of each resource. In Subsection 4.2.4 we will provide a list of criteria that allows us to characterize and compare different FC schemes.

4.2.1 FC Information

The amount of FC information to convey between receiving and transmitting side depends on the FC scheme that is used. In general, feasibility relates to the current downstream buffer state in one way or another. If the absolute buffer state needs to be conveyed, then the FC information is a function of the buffer size B. On the other hand, if only changes in the buffer state have to be reported, then the amount of FC information is constant. In comparison to large buffer sizes, which are for instance required in order to support large RTTs, this translates into significant savings. Also for schemes that signal a start and stop condition, feasibility information consists of just one bit. We will determine the exact amount of FC information I that is required for various schemes in the following Section 4.3. A summary can be found in Table 4.1.

4.2.2 Buffer Size

We have already defined the terms correctness and work-conservation, and were able to associate a certain necessary buffer size to each. We said that for an FC scheme to be correct requires $B_{\min}$, whereas a buffer size of B is needed for it to be work-conserving and correct, $t_c = 0$. When providing less buffer than B, a certain correctness time penalty is to be paid, $t_c > 0$.

There is a relationship between buffer size and performance. As buffer space is expensive and typically scales with $O(N^2)$ for switches, we are inclined towards a scheme that achieves best performance for a minimum buffer size.

As outlined in Chap. 2, the link length between SA and the switch may easily vary, because cables are used instead of a backplane. To account for an additional degree of freedom, we favor an FC scheme that allows the buffer size to be decoupled from the correctness requirement. The decoupling is achieved if the minimum buffer size is constant, $B_{\min} = \text{const.}$, i.e. the buffer size is independent of RTT. This means, the buffer size of an FC scheme will be designed to be work-conserving up to a certain link length. If the link length of a system is increased further, FC shall still provide correctness.

4.2.3 Chip Area

A large portion of chip area certainly is required for buffer space as described in Subsection 4.2.2. However, the implementation of any FC scheme also requires some area. We will define this in the following.

Any FC scheme requires two functions in order to determine the start and the stop condition. It is an accounting and a decision function, and they are realized in an accounting and a decision unit as shown in Figs. 4.2 to 4.6. We will express the area consumption of the schemes in the

number of such units that would conceptually be required for an implementation. We refer to it broadly as *complexity* C, which is sufficient for our purposes. In particular, the complexity of the accounting unit is C_A, while C_D denotes the complexity of the decision unit. The accounting unit is fed by two processes, one that keeps track of packet arrivals and one that accounts for packet departures (also shown in Fig. 4.1). The condition when to start or stop sending packets is determined by comparing the accounting value to a predefined threshold in the decision unit. This threshold T is a parameter.

4.2.4 Summary

As we have seen in Subsection 4.1.2 there is a relationship between performance and buffer size. In particular, by increasing the buffer size, performance can be improved. The time penalties listed in Subsection 4.1.2 in points cα) - cϵ) may be eliminated or hidden. However, there are also other criteria that are necessary to qualify FC schemes, namely the ones related to the implementation costs. We have identified three resources, control path I, buffer size B, and chip size C, that can be associated to implementation costs. In order to find a good trade-off between performance and cost, weights may be introduced according to optimization preferences. In our case, we assign a weight of one to all contributing factors I, B, and C. Therefore, an FC scheme that uses a minimal amount of all these resources is clearly preferred over others. In summary, the quality of an FC scheme is determined by the characteristics:

- Minimum Buffer Size $B_{\min}$. It is an expression of the minimum buffer size to be correct. It is an advantage, if this number is independent of RTT.
- Buffer Size B. It is an expression of the minimum buffer size to be work-conserving and correct. A buffer size of B will eliminate a correctness time penalty, hence in this case $t_c = 0$.
- Amount of FC information to convey I. It may also be used to give an estimate of the expected queueing time t_q.
- Complexity (chip area) C. It is the number of conceptual units required to realize an FC scheme.

4.3 Classification of FC Schemes

As pointed out in Subsection 4.2.3, *any FC scheme requires two functions* to determine the start and the stop condition. These functions are typically realized in distinct functional blocks, or units. Depending on where the two functional units are located, there are two fundamental concepts of how to convey FC information:

1. stateful
2. stateless

In the stateful approach, the accounting state (equivalently buffer state, or buffer occupancy level) of the current node is established or replicated at the upstream node. It means that the accounting state machine is distributed across the two ends of the control path. Therefore, the control path is part of the state machine. As a consequence, the information carried over the link by an FC scheme is either a state change or a buffer-state-related information, i.e. the current buffer state. For this reason we say that the buffer state can be reconstructed at the up-stream

node or – more generally – the FC scheme must be such that the state of the current node is reconstructible at the upstream node.

This is not possible in the stateless approach. Here, the current node processes the state information and derives a decision for the upstream node to obey. The links are not part of the accounting state machine.

For the remainder of this Section, we discuss all the possibilities of accounting and decision unit distributions, and define their properties according to the characteristics derived in Sec. 4.2. Interestingly, all FC schemes that we are aware of are identified by one of these possibilities.

4.3.1 Accounting and Decision at Current Node: The "Grant"-Scheme

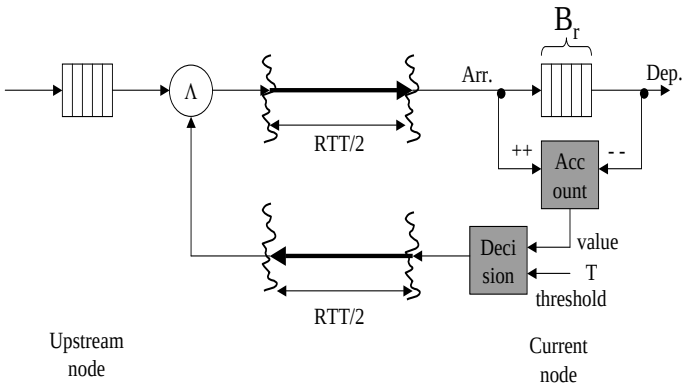


Figure 4.2: Accounting and Decision Unit at the Current Node: Stateless FC Protocol by Grants

If both the accounting and the decision units are located at the current node, we are dealing with a stateless FC protocol. An example of this category is the so-called grant-based FC as for instance described in [12] or [37]. The stateless FC protocol is explained with the help of Fig. 4.2.

Single Threshold – "Grant ST"

The accounting unit keeps track of the current node's buffer state by incrementing the accounting state by one every time a packet arrives. On packet departure the same state counter is decremented by one. If the counter state of the accounting unit is equal to or larger than a given threshold T, then a *stop* signal is generated to prevent the up-stream node from sending further packets and causing buffer overflow. Otherwise, a *go* signal active. For this reason the FC information to be conveyed is a stop or go signal. Once the stop signal is generated, it takes RTT cycles (or $2t_p$ time) until the effect is seen at the current node. For this reason, there needs to be a slack, or excess buffer $B_e = \mathrm{RTT}$ to capture all packets in flight once a stop signal has been generated.

The minimum buffer size $B_{\min} = B_e + T$ is found for a minimum threshold value $T = 1$, and $B_e = \mathrm{RTT}$, yielding $B_{\min} = \mathrm{RTT} + 1 \neq \mathrm{const}$. This also means that if the link is longer than what it has been designed for, correctness is not guaranteed.

The (minimum) buffer size B for being work-conserving and correct is found by considering an additional buffer space of $B_{t_p} = \mathrm{RTT}$, which is necessary to allow for the propagation delays t_p, such that $B = B_e + B_{t_p} = 2\mathrm{RTT}$.

In the minimum buffer size configuration of $B_r = B_{\min} = RTT + 1$, the go threshold is equal to the stop threshold. In general, if the buffer is considered to be successively filled from bottom to top, then the stop threshold needs to be *exactly* RTT locations, and the go threshold *at least* RTT locations away from the top. Duato [11] proposes for a buffer size of $B_r \geq 2\mathrm{RTT}$, the stop threshold to be RTT locations away from the top and to activate the stop upon threshold overstep. The go threshold is further proposed to be RTT locations away from the bottom and to activate a go signal upon threshold understep, which implies two thresholds.

The go and stop signals can be encoded by one bit and hence lead to an amount of FC information of $I = 1$.

The amount of complexity is found to be $C = C_A + C_D = 2$, i.e. the two functional units at the receiving side.

In summary, a grant-based scheme is characterized by these criteria:

- $B_{\min} = \mathrm{RTT} + 1 \neq \mathrm{const.}$
- $B = 2\mathrm{RTT}$
- $I = 1$
- $C = 2$

Multiple Thresholds - Rate-based FC – "Grant MT"

So far, we have considered the case for one threshold ($k = 1$). More generally, if there are $k \geq 1$ thresholds available, then the FC information encodes as $I = \log(k + 1)$. With multiple thresholds, a so-called *rate-based* FC system can be built. The rate-based system allows the buffer state of the current node to be approximated at the up-stream node. In fact, with $k = B - 1$ this scheme becomes a stateful FC scheme. In order to use the approximated state information received, an additional logic block of complexity C_L (at the up-stream node) is required to derive the sending rate. This increases the complexity to $C = C_A + C_D + C_L = 3$. The sending rate is given by the encoding of the thresholds, which in reality corresponds to the area between two thresholds. The area directly relates to a predetermined sending rate. Clearly, the complexity of rate-based systems is higher, in addition to the larger amount of FC information I to convey. An additional optimization of these systems can be achieved, by considering that at most one packet arrives or departs per tu. This simple fact means that at most one threshold may be crossed at any one moment. Therefore, the direction of crossing a threshold would be sufficient information for the up-stream logic to derive the correct sending rate. This optimization reduces the amount of FC information to $I = 2$. The '11' would then for instance mean a threshold overstep, '10' correspondingly an understep, and '0x' would mean no change. The buffer-size requirement for the system to be correct remains $B_{\min} = \mathrm{RTT} + 1$, as well as the requirement of work conservation and correctness $B = 2\mathrm{RTT}$. If the initial sending rate was $r = 1$, and was reduced after the first threshold to a value $0 < r < 1$, then the buffer size is found to be $B > 2\mathrm{RTT}$. If, instead, the rate was "reduced" to $r = 0$ (single threshold scheme), the buffer size to be work-conserving and correct is $B = 2\mathrm{RTT}$. If we do not presume any sending rates, we find a buffer size of $B \geq \mathrm{RTT}$ for being work-conserving and correct.

A rate-based hop-by-hop FC algorithm is for instance considered in [13, 14]. The rate-based FC is a stateless protocol, except for the case of $k = B - 1$ thresholds. In general it is characterized by the following criteria:

- $B_{\min} = \mathrm{RTT} + 1 \neq \mathrm{const.}$
- $B \geq 2\mathrm{RTT}$
- $I = 2$ (optimized); otherwise $I = \log(k+1)$.
- $C = 3$

Optimized Grant – "Grant OT"

A substantial optimization of the scheme in terms of the buffer size as well as the robustness against varying link length is achieved if the number of on-signals slots Q_n is counted over an RTT time window, $T = \sum_{n-\mathrm{RTT}}^{n} Q_n$. This requires an additional complexity of C_L, and increases the complexity to $C = C_A + C_D + C_L = 3$ shown in Fig. 4.3. The maximum number of packets that can be expected at the receiving node within one RTT is T. If $(T + \mathrm{value}) \geq B_r$, a stop signal is generated. For this reason, such a scheme could be operated with $B_{\min} = 1$ provided that at initialization feasibility information is absent at each node. For this particular case, one "on" signal slot generated at the receiving side would result in $T = 1$. In the next cycle, the decision unit would find $T + value = 1$, which is equal to B_r. Hence, the decision unit would generate a "stop" signal for the next RTT cycles. In this configuration the scheme would work correctly, but provoke oscillating go and stop periods. Therefore, the (minimum) buffer size to be correct and work-conserving is $B = \mathrm{RTT}$.

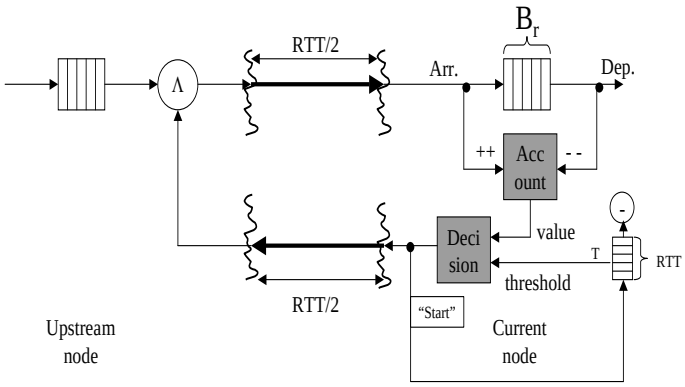


Figure 4.3: Accounting and Decision Unit at the Current Node: Stateless FC Protocol by Optimized Grants

This enhancement was described by Iliadis [28], whereby the expression "$T + value$" is called potential function. The optimized grant is a stateless protocol and is characterized by the following:

- $B_{\min} = 1 = \mathrm{const.}$
- $B = \mathrm{RTT}$
- $I = 1$
- $C = 3$

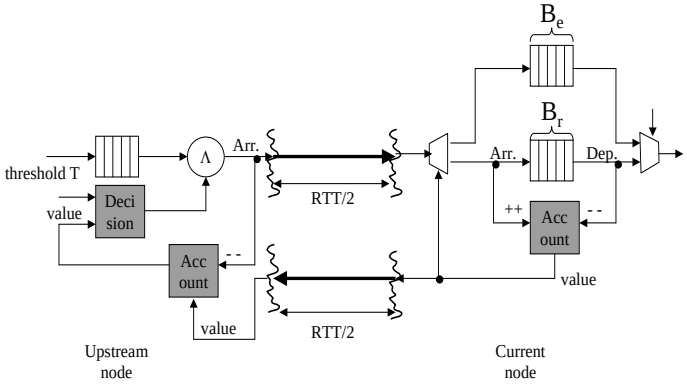


Figure 4.4: Accounting Unit at both the Current and the Up-stream Node, Decision Unit at Up-Stream Node: Stateful FC Protocol by Absolute Credits

4.3.2 Distributed Accounting; Decision at Upstream Node: The Absolute Credit Scheme

Another possibility is to distribute the accounting units over the receiver and the sender, and to move the decision unit to the sender as shown in Figs. 4.4 to 4.6. These arrangements render FC stateful schemes because the current node's buffer state is carried to the up-stream node, hence it is reconstructible.

The receiver accounting unit keeps track of the buffer state as already described. However, the result of the accounting unit is conveyed as the number of free locations to the up-stream node, indicated by *value* in these figures. The up-stream accounting unit is set to this received value, irrespective of the previous counting state[1]. Every time a packet leaves the sender – equivalent to a future arrival at the current node – the state counter is decremented by one. Once the value drops below a certain threshold T in the decision unit, transmission is stopped.

There are three relevant cases to be considered depending on the threshold level T and the frequency of updating the counter state. The first one explains the simplest case of absolute credits ("Credit Abs"). The second is an optimization to reduce the amount of FC information I to convey ("Credit Abs I"), while the third achieves an additional buffer reduction ("Credit Abs B").

Absolute Credits - Simplest Case – "Credit Abs"

The counter state may be transmitted in *every packet cycle*. By following this approach, the decrementing process at the up-stream node as shown in Fig. 4.4 is in fact superfluous. Let us assume that the accounting unit of B_r sends a $value = 0$, which means no more room to accept any further packets in B_r. Let us further assume it has been larger than zero in the previous RTT packet cycles, and the output is blocked. Then the receiver must be able to catch all, but not more than RTT packets in flight. As B_r is completely filled with packets, an excess buffer B_e is required to receive the outstanding packets. As shown in Fig. 4.4, the packet stream is switched to B_e for $value = 0$. This arrangement is therefore correct. The minimum buffer size is $B_{\min} = B_{r,\min} + B_e = 1 + \mathrm{RTT}$. A buffer size of $B = 2\mathrm{RTT}$ is required for the scheme to be work-conserving and correct. Owing to the excess buffer B_e, the first $value > 0$ is sent as soon as B_e is empty. Therefore, the worst-case restart time is $t_r = \mathrm{RTT}$. The FC information

[1]This makes the scheme robust against errors in FC information at the cost of complexity and amount of FC information

is the encoding of $value$, hence $I = \log(B_r) + 1$. The complexity C is easily found to be $C = 2C_A + C_D = 3$.

In reality the FC scheme of Fig. 4.4 is operated such that the counter update is performed on every *packet departure*. This also implies that a counter value of zero is not returned, which has no consequences on either the amount of FC information I to convey or the complexity, but on the buffer size.

The minimum buffer size necessary to be correct then becomes $B_{\min} = B_{r,\min} + B_e = 1$, whereas $B_{r,\min} = 1$ and $B_e = \min(B_r - 1, \mathrm{RTT} - 1) = 0$ for the following reason. For values of $B_r < \mathrm{RTT}$ there can be no more than B_r packets consecutively scheduled at the up-stream node. Again, on every packet departure the up-stream counter is decremented by one and transmission is stopped as soon as the counter reaches $T = 0$. The first packet of a sequence of B_r packets triggers the return of a state value of B_r provided the reception buffer is empty and the node was unblocked. All subsequent $B_r - 1$ packets may be blocked at the output of the current node output and are stored in the receive buffer. Hence, they do not trigger a value update. The up-stream node can not send packets for the remaining $RTT - B_r$ cycles because the feasibility information is missing, which by the way, is the correction time penalty. Upon reception of the state value of B_r mentioned earlier, permission is given to send this number of packets. These packets will then go into the excess buffer B_e, which needs to provide space for $B_e = B_r - 1$ packets, because the first packet will still fit into B_r. The special case of $B_r = B_{r,\min} = 1$ delivers $B_e = 0$. For values of $B_r >= \mathrm{RTT}$, the same reasoning would result in a required slack buffer of $B_e = \mathrm{RTT} - 1$. As long as $B_e = B_r$, an increase in link length does not require an increase in buffer size to remain correct.

The (minimum) buffer size to be work-conserving and correct becomes $B = B_r + B_e = \mathrm{RTT} + \mathrm{RTT} - 1 = 2\mathrm{RTT} - 1$. The restart time t_r is as large as the excess buffer, because it needs to be drained before the first state larger than zero is transmitted.

In summary, the simplest case of absolute credits is characterized as follows:

- $B_{\min} = B_{r,min} + B_e = \mathrm{const.}$ for $B_e = B_r$
- $B = 2\mathrm{RTT} - 1$
- $I = \log B_r + 1$
- $C = 3$

Controlling the FC Bandwidth – "Credit Abs I"

The overall FC information can be reduced to $I = \frac{\log(B_r+B_n)+1}{n}$ if the counter state is updated on every nth packet departure. In this case, the up-stream state and its decrementing process ensures that in the absence of the latest counter state, the total receive buffer $B_{\mathrm{tot}} = B_r + B_n + B_e$, as shown in Fig. 4.5 does not overflow. We recall that the transmitted counter state is the number of free locations at the receive node. Sending FC information every nth cycle signifies an intentional queueing delay t_n, which will impact performance. This impact can be hidden by an additional buffer space B_n, worth n packets as already described in Subsection 4.1.2. Complexity is increased to $C = 2C_A + 2C_D = 4$ because of an additional decision unit required to determine the point in time of returning a FC message. This unit is subtitled "bandwidth control" in Fig. 4.5 as it is able to control FC bandwidth. As a result the amount of FC information to convey can be reduced at the cost of additional complexity and buffer space.

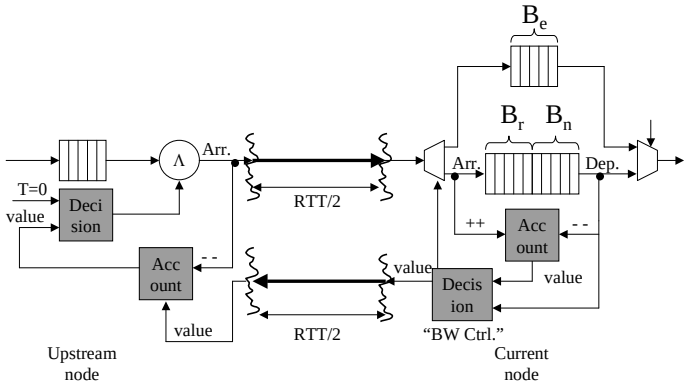


Figure 4.5: Accounting Unit at both the Current and the Up-stream node, Decision Unit at Up-Stream Node: Stateful FC Protocol by Absolute Credit Update – FC Bandwidth Optimized

This scheme has been described in [8] and is known as the N123 scheme. Buffer space N1 corresponds to B_e; B_r finds its counterpart in N3, and B_n in N2.

The improvement over the simple case of credits is achieved in the reduction of the amount of FC information I to convey. This comes at the cost of additional complexity and buffer space B_n.

- $B_{\min} = 1 = \text{const.}$
- $B = 2\text{RTT} - 1 + B_n$;
- $I = \frac{\log(B_r + B_n) + 1}{n}$
- $C = 4$

Eliminating the Excess Buffer B_e – "Credit Abs B"

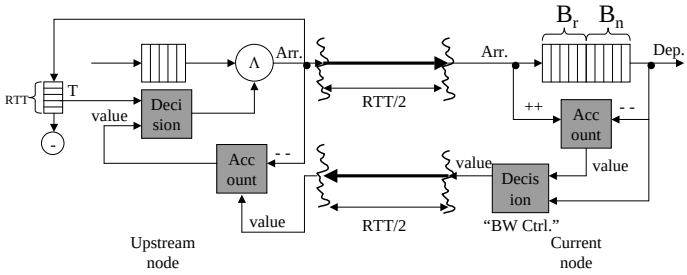


Figure 4.6: Accounting Unit at both the Current and the Up-stream node, Decision Unit at Up-Stream Node: Stateful FC Protocol by Absolute Credit Update – Buffer and FC Bandwidth Optimized

If the number of departures at the up-stream node are counted over the last RTT time period, the buffer requirements can be reduced by the excess buffer B_e to $B = B_r + B_n$ as shown in Fig. 4.6. The correctness and work-conserving requirements are fulfilled at the same time. This scheme is work-conserving for $B_r = \text{RTT}$. It is correct, because "*value*" equals the amount of receive buffer space that is still available, without accounting for the packet arrivals during the last RTT time-period, similar as in the two previous cases, whereas Q_n represents exactly this number. Therefore, transmission at the up-stream node is stopped if $T \geq \text{value}$. An additional

unit is required to realize the calculation of T, which increases the complexity by C_L. Together with the reduced amount of FC information to convey, as already described previously, the complexity there is $C = 2C_A + 2C_D + C_L = 5$.

In literature, this enhancement is referred to as the N23 scheme [8]. Note that the same idea of counting a certain type of events over a period of RTT cycles was also applied by Iliadis [28].

Together with the FC information reduction of the previous subsection, the last scheme is characterized as follows:

- $B_{\min} = 1 = \text{const.}$
- $B = \text{RTT} - 1 + B_n$
- $I = \frac{\log(B_r + B_n) + 1}{n}$
- $C = 5$

4.3.3 Accounting and Decision at the Up-stream Node: The Relative Credit

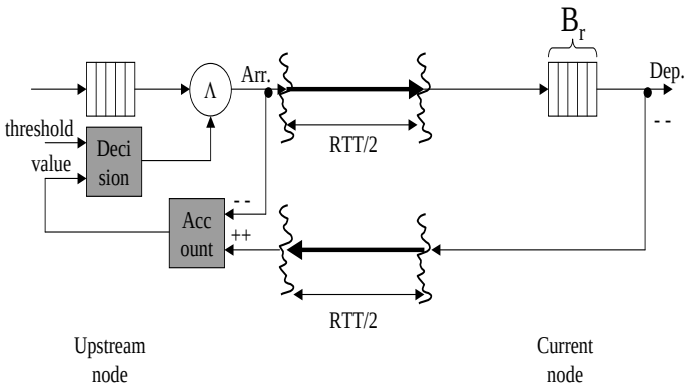


Figure 4.7: Accounting and Decision Unit at the Up-stream Node: Stateful FC Protocol by Relative Credit Update

A third way ("Credit Rel") to distribute the basic units between sender and receiver is to place them both at the up-stream node, as shown in Fig. 4.7. We are dealing with a stateful protocol also in this case , because the sender is aware of the receiver's buffer state. The receiver's buffer state happens to be counted at the sender side, and the FC scheme hence conveys state changes. Each packet departure at the current node is returned to the up-stream node as a permission ("*credit*") to send one more packet. Such a scheme is identified as a relative credit update FC protocol. Upon each up-stream packet departure – a future arrival at the current node – the counter state is decremented by one. If the counter equals zero, packet transmission is stopped. In this way it is ensured that the receive buffer does not overflow. Therefore, correctness can be assured with $B_r = 1$, yielding a minimum buffer size of $B_{\min} = 1$. The (minimum) buffer size to be work-conserving and correct is $B = \text{RTT}$. This scheme has a restart time $t_r = 0$ because a credit is generated immediately upon every packet departure, . The amount of information is $I = 1$, to indicate one credit. Its complexity is $C = C_A + C_D = 2$.

The relative credit protocol is described and used for instance in [10, 1, 2]. Its properties are briefly summarized:

- $B_{\min} = 1 = \text{const.}$
- $B = \text{RTT}$
- $I = 1$
- $C = 2$

4.4 Support of Traffic Classes

The following needs to be considered when assigning QoS to flows, such as priorities or lanes. Let us assume our system supports p priorities and l lanes. Higher-priority traffic in a lane may overtake lower-priority traffic. Lanes are independent of each other, and allow the assignment of certain bandwidth or delay guarantees. l lanes allow l virtual networks to co-exist in one SF. In communication environments, this feature can be used to support different independent traffic flows at a finer granularity, for instance, an OC-768 flow that consists of four OC-192 flows. The lane concept may allow the four OC-192 flows to be switched independently of each other. In multiprocessor environment, lanes are important to separate request from reply messages, in order to avoid cyclic dependencies ("dead-locks").

In a stateless protocol the amount of information to send increases to $I = 1 + \log(p) + \log(l)$, because the current node needs to inform the sender as to which priority/lane combination is allowed to make forward progress without loss of packets. The necessity of go/stop signals per QoS class is explained as follows: let us consider a priority scheme, in which the highest priority always takes precedence over lower priorities. Assume there was only one go/stop signal available for all priorities. The traffic scenario is that the current node is blocked/contended for low-priority traffic and there is only low-priority traffic available at the up-stream node. The current node fills up completely until the stop signal is generated. If at the same time as the stop signal arrives at the upstream node a high-priority packet arrives, it may not proceed, even though there are resources available at the downstream node. With go/stop signals per priority, this could have been avoided by setting appropriate thresholds at the current node and hence shutting off the arrival process of lower-priority packets in time and leaving buffer space for higher-priority traffic.

Additionally, an upstream arbiter is needed in order to determine which packet to send if there are multiple go signals for different priorities/lanes available at the same time. This increases the complexity of those schemes.

This process is fundamentally different in stateful protocols, where it is sufficient to update the sender's accounting unit according to the scheme utilized. The sender is aware of the receiver's buffer state and may therefore stop sending low-priority packets on a certain buffer state at its own behalf. Buffer hogging by low-priority packets as described in the example above is thus prevented. The same reasoning applies to lanes. The complexity in stateful schemes is also increased because the up-stream node needs to arbitrate for link access among competing sources according to the counter state.

4.5 Summary

We have presented and discussed the different possibilities of distributing the accounting and decision units across sender and receiver. This approach allowed us to identify stateless and stateful FC schemes and provided the framework to describe relevant FC schemes. Before

increasing the number of dimension of our basic FC problem in the next chapter, we want to summarize and compare the combinations presented. This comparison is found in Table. 4.1.

Table 4.1: Comparison of FC Schemes

FC scheme	B	$B_{\min}$	I	C	class
Grant ST	$2\mathrm{RTT}$	$\neq \mathrm{const.}$	1	2	stateless
Grant MT	$\geq 2\mathrm{RTT}$	$\neq \mathrm{const.}$	2	3	stateless
Grant OT	RTT	$\mathrm{const.}$	1	3	stateless
Credit Abs	$2\mathrm{RTT}-1$	$\mathrm{const.}$	$\log(B_r)+1$	3	stateful
Credit Abs I	$2\mathrm{RTT}-1+B_n$	$\mathrm{const.}$	$\frac{\log(B_r+B_n)+1}{n}$	4	stateful
Credit Abs B	$\mathrm{RTT}-1+B_n$	$\mathrm{const.}$	$\frac{\log(B_r+B_n)+1}{n}$	5	stateful
Credit Rel	RTT	$\mathrm{const.}$	1	2	stateful

From this list, we identify the "relative Credit" scheme as the one using the least amount of resources. It has a complexity of 2, needs an amount of information $I = 1$, and does not have a restart time, hence $B = \mathrm{RTT}$. Also its correctness is independent of the buffer size. The scheme that comes closest to it is the "grant Optimized", which has the same parameters, except that it has a complexity of 3. Also, the optimized grant scheme is a stateless protocol, which would require to increase the amount of FC information to convey in the case of QoS support. This was a result of Sec. 4.4, where it was found that only in stateless protocols is the support of possible QoS requirements reflected in the amount of FC information to convey. Table 4.1 further shows the relationship between buffer size and RTT.

Chapter 5

The N-dimensional Flow Control Problem

We have looked at the basic FC problem, which is about interconnecting two buffers efficiently. We have derived several quality characteristics that allow us to categorize and evaluate existing solutions. We now increase the number of buffers by N at either side of the link while keeping the bandwidth of the data path. We will consider all resulting configurations, 1×1 (as a recapitulation) in Section 5.2, $1 \times N$ in Section 5.3, $N \times 1$ in Section 5.4, and $N \times N$ in Section 5.5. Each such configuration is also referred to as a *FC domain* (FCD). For a pairwise interconnection as encountered in the $N \times N$ problem, each of the N pairs is also referred to as *FC subdomain* (FCSD).

These combinations occur as FC problems at various levels in a packet switched network. This is shown in Fig 5.1 at the example of a networking node in a communications environment, which is an unfolded system view as introduced in Sec. 2.3. The typical case for a multiprocessor environment can easily be derived as we have also shown in Chap. 2, if the SAs and the NPs are combined into one single entity, a processing node containing the network interface.

According to the OSI layered model of protocol architecture, we identify as our layers of interest the transport layer that operates between end-points (DTE's), the network layer, that provides upper layer with independence from data transmission and switching technologies, and the data link layer, that performs transfer across physical links. Since we only look at the FC specifics of such layers, we differentiate ourselves by saying these protocols to operate at a certain level, which is relevant to the corresponding layer. In Figure 5.1 we consequently show the FC-specifics of the transport layer, the transport-level, of the network layer, the entry-to-exit level, and of the data link-layer which comes in two flavors, the network access (NA)-level and the link-level. This distinction is made in order to better visualize the location of the level in the communication subsystem.

The aim of this chapter is to

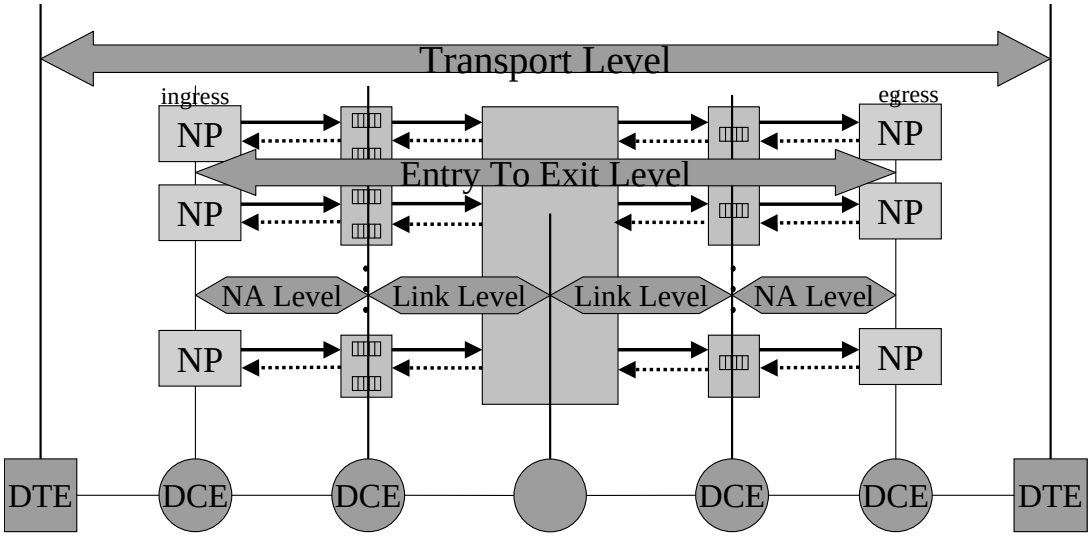
1. present the cost of FC information in switches
2. generally discuss the different arrangements in the light of stateful and stateless FC classes. On this behalf, we are interested in:
   (a) complexity; we want to see how complexity C as introduced in Subsection 4.2.3 changes.
   (b) FC information; we want to see how FC information I as introduced in Section 4.1 and 5.1 is influenced.
   (c) FC bandwidth; unlike the *data path*, most systems scale the *control path* bandwidth,

when growing to N buffers at either side of the link. We want to investigate, whether this is a requirement or not.

(d) performance arguments; we want to collect arguments towards a performance evaluation of such systems that we will need for the construction of an appropriate reception scheduler strategy.

3. give some examples where such FC arrangements occur in the real world and how they would map onto the different FC levels according to Fig. 5.1.

4. see where in switching such problems are addressed and list corresponding examples. We want to investigate how the FC problem between adapters and the switch, i.e. the link-level FC, has been addressed and solved in known switch architectures.



DTE: Data Terminating Equipment
DCE: Data Communications Equipment

Figure 5.1: Flow Control Levels

While the first and the last point will be discussed in separate sections (Section 5.1 and 5.6) the other two points will be addressed in the subsections of the resulting configurations.

In all cases we exclude speed-up, i.e. one single linked source transmits one packet per tu. We further exclude different classes of service ("QoS"), such that N relates to destinations or sources, but not to traffic classes. We define the complexity C_{xy} to be the complexity C for $x \times y$ connections. We express the resulting complexity in units of the base complexity C_{11} that is given for individual accounting and decision unit distributions in Table 4.1. As a reminder, we would like to point out that the amount of flow control information I to convey is always counted without considering QoS aspects. In particular, for stateless FC schemes, I changes for different QoS requirements as outlined in Section 4.4.

5.1 The Cost of FC Information

Typically, the FC bandwidth I is a function of N. It could be a vector representing the state of all N relevant buffer resources in the worst case[1], or the encoding of $n, 1 \leq n \leq \frac{N}{\log(N)}$ outputs,

[1] According to our assumption of excluding QoS aspects and multicast

yielding $I = n \log(N)$. In the worst case, the total amount of FC information to convey is $I_{wc} = N$, while in the best case it is $I_{\mathrm{bc}} = \log(N)$.

There are two ways to send FC information. The first one is to send it *in-band*, that is, incorporated into the data network, the second one is *out-of-band*, that is, over a dedicated network, i.e. links. Sending FC in-band requires a speed-up in order to sustain the targeted bandwidth. On the other hand, the out-of-band approach requires an increase in the number of links.

Thus, both ways induce a certain overhead which directly translates into *additional link cost*. Additional link cost manifests itself not only in the possible development of a faster link technology, but also in larger real estate on the chip and higher power consumption during operation. As link cost dominates the total switch cost, avoiding additional link cost will have an impact on the total switch cost. This is in particular true, as our calculations are for one single link, whereas a SF requires $2N$ links in total. We are able to control this portion by the amount of FC information to convey. It is also important to bear in mind that future generations of switches will support a larger number of ports than the current switches. Because the additional link cost increases with I, which itself increases with the number of ports, additional link cost will become more and more relevant.

The additional link cost and its dependency on the number of switch ports N calculates with the cost model shown in Fig. 5.2. The total cost K is composed of the base cost K_b, i.e. the cost to support the targeted bandwidth, and the additional cost K_{add} that is necessary to account for the FC overhead. According to our assumption of Chap. 2 of a slotted network, the bandwidth may be expressed in packet size P [bits] per packet cycle, yielding $K_b \sim P$. The FC overhead is expressed in the amount of FC information I, and hence the additional link cost is proportional to the amount of FC information: $K_{\mathrm{add}} \sim I$. Starting from a required speed-up S, we can say that

$$K = SK_b, \tag{5.1}$$

whereby $S = 1 + \frac{K_{\mathrm{add}}}{K_b}$, which can be also expressed as

$$S = 1 + \frac{I}{P}. \tag{5.2}$$

Therefore the increase in link cost due to the FC overhead is attributed to the term I/P.

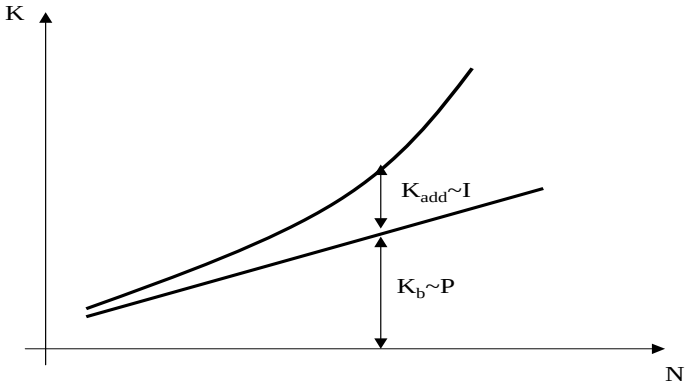


Figure 5.2: Cost Model

This relationship is concretely shown for three different packet sizes, 32, 64, and 128 bytes, in Fig. 5.3, as well as the worst and best case amount of FC information $I = N, \log(N)$. While smaller packet sizes and smaller switch sizes are more relevant to multiprocessor environments

(area for same I between $P = 32$ and 64 curves), larger packet sizes and larger switch sizes are more relevant to communication systems (area for same I between $P = 64$ and 128 curves). For the minimum amount of FC information, the figure clearly demonstrates that FC will become a bottleneck for larger switch sizes as it scales with $O(N)$, unless the amount of FC information is minimized such that it only scales with $O(\log N)$. The achievable savings in link cost amounts to the difference between $I = N$ and $I = \log(N)$ curves. As an example, for a packet size of $P = 64$ bytes, the savings are as high as 5% (for each link) for $N = 32$, reach 10% for $N = 64$, and are close to 25% for $N = 128$. For packet sizes of $P = 32$ bytes, the same savings are achieved for $N = 16$, $N = 32$, and $N = 64$.

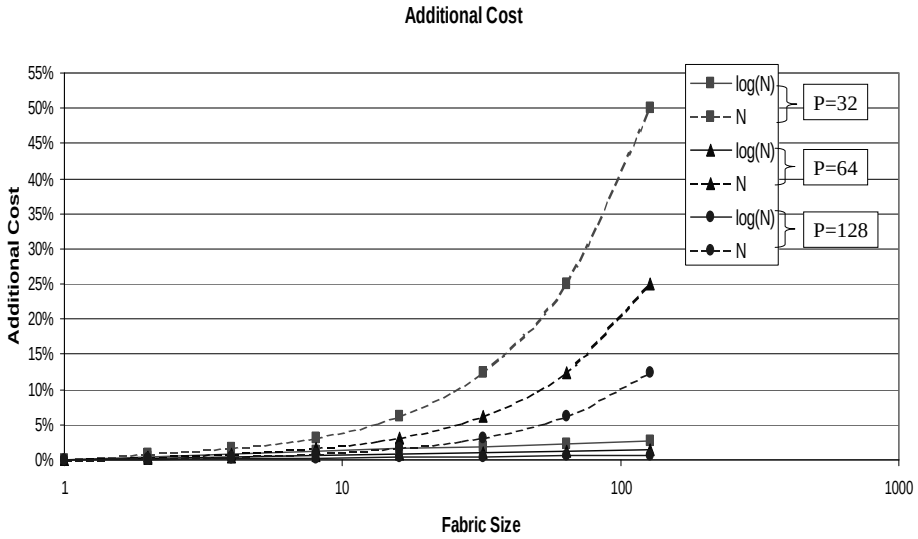


Figure 5.3: Projection of the Per-link FC Cost

We therefore conclude that having an FC scheme that allows us to operate with the minimum amount of FC information is crucial, especially to prevent FC becoming a bottleneck in current and future switch development. Figure 5.3 indicates that significant savings can be achieved for communication switches (larger packet sizes) with port sizes of 32 and larger, and for multiprocessor switches (smaller packet sizes) and port sizes of 16 and larger.

The aspect of minimum FC information is closely related to the queueing time t_q, resulting in performance issues that we will treat thoroughly in Chapter 7.

5.2 Point-to-Point Connections (1×1)

This case has already been treated in the last chapter. It lead us to distinguish two basic classes of FC schemes, stateful, and stateless. In the following we will give some real world examples, where point-to-point FC issues are found.

5.2.1 Examples

The 1×1 FC problem is found at the data link or network layer, as well as at the transport layer, when considering the OSI model. While the former two involve a point-to-point connection of neighboring nodes, the latter includes multiple hops.

A hop-by-hop FC scheme may be found at a link level, or at a NA-level according to Fig. 5.1. The end-to-end FC problem is often mapped onto the transport level.

A 1×1 FC loop is typically enhanced by returning information about in-order and error-free reception. In such cases, the scheme is referred to as "window-based" FC. It involves a sequence number and an error detection/correction-code per packet. The main idea is to acknowledge the correct reception of data to the sender. Acknowledgements are piggybacked or sent as extra packets as positive, negative, or cumulative feedback.

Obviously, the complexity of window-based FC schemes is higher than the basic FC scheme presented earlier, because it requires an acknowledgement dispatch unit at the receiver and an acknowledgement control unit at the sender. It further needs to implement a retransmission strategy, such as GO-back-N, or selective repeat, for cases that packets were received in error [2]. Window-based FC is a technique to achieve reliable packet transmission. It is used for instance in HDLC [29] at a link-level, or in TCP at an end-to-end level [18, 19].

Often, the granularity of packets is subdivided into smaller entities on which FC is applied. These entities are referred to as **fl**ow control d**ig**i**ts** (flits) [84]. The advantage being that buffers need only to be as large as to hold a couple of flits instead of entire packets, which would be required in the alternative packet store-and-forward approach. The first flit out of a sequence of flits that make a packet determines the destination, and hence carries the packet overhead. All subsequent flits follow until an end flit is detected. Moreover, this technique allows to reduce the average packet delay through the network of a system. Packets may not have to wait for the correct reception of its entirety before proceeding at intermediate nodes, but may do so once a smaller flit entity has been received. Also, a second packet does not have to wait for the entire first packet to be transmitted, before it may proceed on a given channel. This technique is also known in literature as wormhole switching (or wormhole routing). Wormhole switching is used in numerous multiprocessor environments such as in the SP-2. It has the disadvantage that a blocked worm easily spreads across the network, hence blocking resources that could be used by other flows.

In terms of reliable delivery of packets, flit based FC allows to check for correct transmission on a per-hop basis, for instance by a window-based scheme. This is why it is said to also achieve better link (channel) utilization, because it reduces the chances of end-to-end packet re-transmission significantly.

While the window-based FC is a solution to the requirement of reliable delivery of packets, flit-based FC addresses the packet latency aspect in - especially - multi-processor environments. The FC scheme that we study can easily be enhanced with both aspects. It is mainly a matter of the segmentation and reassembly function implemented in the SA as pointed out in Subsection 2.1.3.

5.2.2 Summary

The complexity C_{11} and the amount of FC information I are summarized for different FC schemes in Table 4.1. In Section 4.5, we have selected as representative for the stateless FC class the optimized grant, and for the stateful FC class the relative credit FC protocol. For these two the amount of FC information I is $I = 1$.

Obviously, for 1×1 connections there is no requirement to return more than one FC event per packet cycle, as there is at most one arrival and one departure.

The performance aspects, in particular work-conservation was discussed in the Chapter of the basic FC problem (Chapter 4).

[2] errors are: bit error(s), sequence errors, packet duplication or dropping

We have seen this problem to be present at all three levels, namely link, network, and transport level. We have further seen that the 1×1 FC problem is often enhanced by a "window mechanism" to achieve reliable delivery of packet. Furthermore, a finer granularity at which FC is applied often reduces the utilization of resources and results in better network performance. We concluded that our FC discussion and results are orthogonal to these issues. Our FC schemes derived later can be enhanced by both aspects.

5.3 Point-to-Multipoint Connections ($1 \times N$)

5.3.1 General

There is one buffer that sends packets to N distinguished destinations, which is shown in Fig. 5.4. The arrival rate λ and departure rate $\mu_1 \ldots \mu_N$ are shown to indicate that we are dealing with random processes. This is also true for the subsequent Figs. 5.5 and 5.6 of the $N \times 1$ and $N \times N$ FC problem. Since we exclude speed-up, there is at most one arrival to any of the N destinations per packet cycle. On the other hand, there can be up to N *parallel departures*.

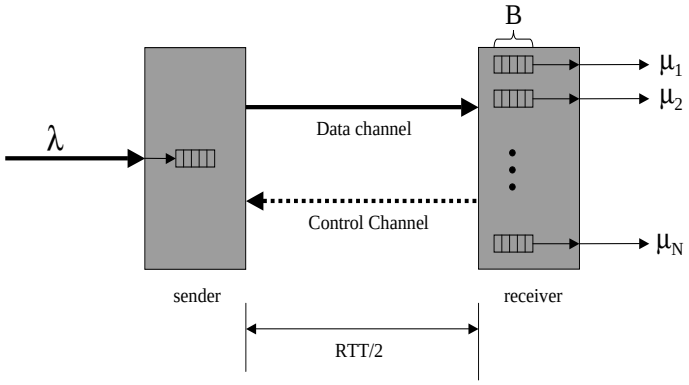


Figure 5.4: The $1 \times N$ FC Problem

For this arrangement using a *stateless* FC scheme, a packet arrival may cause a threshold overstep at any of the destinations. Since there is at most one packet arrival per tu, at most one stop signal may have to be generated per tu as well. For correctness reasons a triggered stop-signal *must* be sent in the next cycle, unless some extra buffer space B_n is provided, as introduced in Subsection 4.1.2, which we do not want to take into account here. Therefore, there is a requirement to return all "must send" signals within the next tu. However, as there is at most one such signal generated per tu, the requirement on FC bandwidth is to return at least one FC event. On the other hand there are $n, 0 \leq n \leq N$ possible (concurrent) departures. Here we note that n possibly triggered go-signals *can* be returned in the next cycle. There is no necessity for correctness reasons to return all n within one tu. Still, for work-conservation reasons (performance) and stability, at least one has to be returned in the next cycle. The minimum bandwidth that must be provided for the return channel is therefore the capability to return one FC-event per packet cycle.

Also, it could for instance occur that packets are departing from multiple buffers and may for this reason change to a "go" state. At the same time another buffer had a change to a "stop" state. In such a case, it must be assured that the "stop" information gets precedence over the "go" information instances (feasibility information), in case the reverse path is confined in bandwidth. This naturally causes an extra delay of feasibility information. For the generation

of multiple feasibility information instances at the same time, some will be delayed more than others.

A delay of feasibility information may harm performance. If a packet waiting at the head of the send queue receives go-signals for different destinations than its own, it may not proceed, i.e. it is temporarily blocked. If the corresponding go-signal could have been sent instead and possibly earlier, better performance was achieved, even though the overall performance of this arrangement is limited by the HOL-blocking phenomenon. We conclude that ordering the return of FC events has an impact on performance, which is an important observation that we will come back to later.

The amount of FC information is found to be $I = n(\log N + 1)$, $1 \leq n < \frac{N}{\log N}$ because the sender must be able to identify the go, and stop signal respectively, and must be able to associate the related send buffer ($\log N$). n such information units can be returned in parallel. For $n \geq \frac{N}{\log N}$, sending a bit-vector of length $I = N$ is more efficient, whereby each bit position corresponds to a destination buffer and '1' for instance signifies a "go" and a '0' a "stop" signal.

The complexity is found to be $C_{1N} = N(C_{11} + U) + \mathrm{RXS}$, because each reception buffer maintains its own state and decision unit and the sender maintains a unit U per destination to remember the last send state (go/stop) until further notice from the receiver. These units U were not necessary provided a FC bandwidth of N FC-events per tu. Moreover, there is a unit RXS required at the receiver that decides which go-signals from which destination gets access to the reverse path. It must further be able to give precedence to the "must send" operations. Such a unit is for instance shown in Fig. 7.23 that explains the $N \times N$ FC problem in more detail. It is referred to as the *reception scheduler (*RXS*)* for reasons explained later.

If there is no FC bandwidth restriction, then the complexity is found to be $C_{1N} = NC_{11}$, because the go/stop information was constantly refreshed and therefore remembering the send state, and the RXS was not needed. On the other hand I would then need to be $I = N$.

For *stateful* FC schemes a packet arrival does not cause a FC event, because the state change has already been covered for at the upstream node. This is, as we recall, because the accounting of the reception buffer is done upstream. All feasibility information instances created by n simultaneous departures *can* be returned in the next cycle. For a bandwidth confined return channel a RXS unit was needed to order the return of feasibility information. Therefore, complexity here is found to be $C_{1N} = NC_{11} + \mathrm{RXS}$, because each of the receiving buffers must be reconstructible at the sending side, which requires N accounting and decision units to be replicated at the sender.

The amount of FC information is found to be $I = n(\log N)$, $1 \leq n \leq \frac{N}{\log N}$ because the sender must be able to return feasibility information to the related send buffer ($\log N$). n such information units can be returned in parallel. For $n > \frac{N}{\log N}$, sending a bit-vector of length $I = N$ is more efficient, whereby each bit position corresponds to a destination buffer and '1' for instance signifies a feasibility information and a '0' has no meaning.

In Table 5.1, we have summarized the differences in complexity of stateless vs. stateful schemes in the light of amount of FC information I to convey.

It is important to see that a stateless FC scheme has "must send" and "can send" requirements, while the stateful approach has "can send" requirements only. If a "must send" requirement is violated, then so is correctness. The "must send" requirement is violated, if the stop signal is not sent the next cycle after its detection. The "can send" cases signify that all the information may be sent at once, but this is not stringent. If all "can send" (feasibility) information is not sent at once, for instance due to a bandwidth-confined reverse channel, queuing time

Table 5.1: $1 \times N$ FC Problem Complexity C_{1N} for Stateless and Stateful FC Classes and the Required Amount of FC Information I.

Criteria	stateless	stateful
$I < N$	$N(C_{11} + U) + \mathrm{RXS}$	$NC_{11} + \mathrm{RXS}$
$I = N$	NC_{11}	NC_{11}

t_q of feasibility information occurs, possibly leading to a performance impact. This has not been quantified yet. Furthermore, it has not been studied, to what extent such parallel departures are to be expected. Clearly, if they do occur, they will lead to contention of feasibility information. These problems will be studied in detail in another chapter, however for $N \times N$ connections.

5.3.2 Examples

This combination is found typically at a NA-level according to Fig. 5.1. Its objective is to throttle external traffic to prevent overall internal buffer congestion. An example would be the CSIX specification [25].

5.3.3 Summary

The FC complexity C_{1N} and the amount of FC information I required are summarized in Table 5.1. The minimum number of FC events that have to be returned in the next packet cycle is one for both stateless and stateful FC schemes. By argumentation, we identified the order of returning feasibility information to have a performance impact, an issue which has been to the best of our knowledge never considered in literature. The $1 \times N$ FC problem is mainly found at a NA level.

5.4 Multipoint-to-Point Connections ($N \times 1$)

5.4.1 General

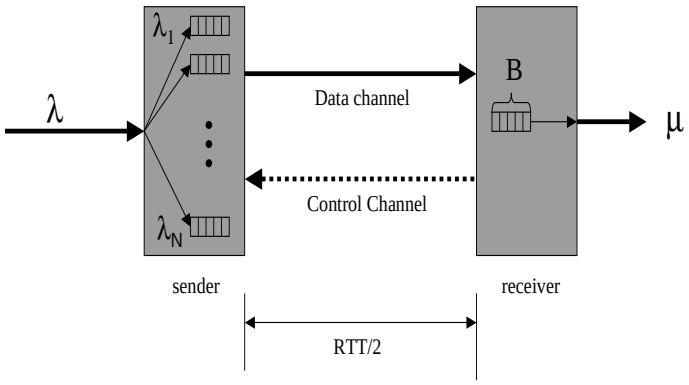


Figure 5.5: The $N \times 1$ FC Problem

There are N buffers that all send packets to one single receive buffer, as shown in Fig. 5.5. The system is characterized by at most one arrival and one departure per tu.

For *stateless* FC, one feasibility information stops and restarts all sending sources ($I = 1$), because there is only one accounting and decision unit, which is located at the receiver. The complexity is found to be $C_{N1} = C_{11}$.

For *stateful* FC schemes, the sender must be able to find out whether the feasibility information is meant for him or not ($\log N$). As there are no parallel departures, merely at most one, the amount of FC information is written as $I = \log N$. Here, the complexity is $C_{N1} = NC_{11}$, because each send queue maintains its own accounting and decision unit. The complexity in the light of FC information I is summarized in table 5.2.

Table 5.2: $N \times 1$ FC Problem Complexity C_{N1} for Stateless and Stateful FC Classes and the Required Amount of FC Information I.

Criteria	stateless	stateful
$I = 1$	C_{11}	—
$1 < I < N$	—	NC_{11}

Since multiple sources contend for link access an arbiter must be in place to resolve the problem. The arbiter might for instance employ a RR , or a weighted round robin (WRR) scheme. The $N \times 1$ is not a real problem from the FC perspective, as the maximum FC bandwidth required is one FC event per tu. This arrangement is a typical problem of the data path and numerous solutions for better arbitration have already been proposed.

5.4.2 Examples

The combination $N \times 1$ is typically found in entry-to-exit level FC, as shown in Fig. 5.1. Its objective is to limit the flow between specific source destination pairs to prevent congestion at the destination. An example would be SNA (System Network Architecture) virtual route pacing control, where the entry node must obtain permission from the exit node, depending on its buffer availability, before sending a new group of k packets, k being a window size [16].

Some network implementations do not have an entry-to-exit level protocol. As an alternative, DiffServ for instance in IP networks operates a (static) Random Early Discard (RED) [22], or (dynamic) bandwidth allocation technology (BAT) algorithm [21] at the egress.

In general, the $N \times 1$ FC problem is found at queueing points along the data path, whenever multiple queues contend for link access. Examples of arbitration mechanisms are for example found in [104].

5.4.3 Summary

The complexity C_{N1} in dependency of the amount of FC information I is listed in Table 5.2. As in this combination there are no parallel departures, there is no requirement, to return more than one FC event per tu for both stateless and stateful schemes. From a performance perspective, the $N \times 1$ FC problem is typically found along the data path, when multiple flows merge to one. Here it is the responsibility of sophisticated data path schedulers, i.e. schedulers at the sender, to achieve good performance according to QoS characteristics. The FC channel has here no influential power. The $N \times 1$ FC problem is for instance found at an entry-to-exit level.

5.5 Multipoint-to-Multipoint Connections ($N \times N$)

5.5.1 General

There are N sources connected to N destinations, each one representing a source-destination pair, sharing a common link. This arrangement, shown in Fig. 5.6 is characterized by the receiving side experiencing at most one arrival, but up to N concurrent departures per tu.

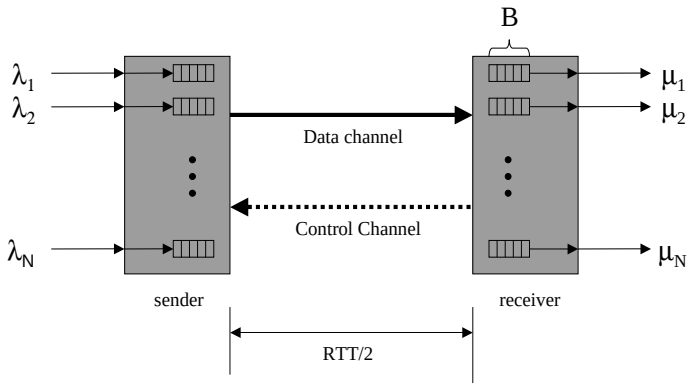


Figure 5.6: The $N \times N$ FC Problem

For a *stateless* FC scheme, this single arrival might be the reason for the issuance of a stop signal, which is a "must send" operation. Up to N parallel departures translate into "can send" operations. The amount of FC information is found to be $I = n(\log N + 1)$, $1 \leq n < \frac{N}{\log N}$ because the sender must be able to identify the go, and stop signal respectively, and must be able to associate the related send buffer ($\log N$). n such information units can be returned in parallel. For $n \geq \frac{N}{\log N}$, sending a bit-vector of length $I = N$ is more efficient, whereby each bit position corresponds to a destination buffer and '1' for instance signifies a "go" and a '0' a "stop" signal.

The complexity is found to be $C_{NN} = N(C_{11} + U) + \mathrm{RXS}$, because each reception buffer maintains its own state and decision unit and each sending source maintains a unit U that allows to remember the last send state (go or stop) per source. Moreover, there is a unit RXS required at the receiver that decides which go-signals from which destination gets access to the reverse path. It must further be able to give precedence to the "must send" operations. If there is no FC bandwidth restriction, then a reduced complexity $C_{NN} = NC_{11}$ is found at the cost of increased FC information $I = N$.

For *stateful* FC schemes there are simply up to N departures to be returned as credits per tu. Hence, the amount of FC information is found to be $I = n \log N$, $1 \leq n \leq \frac{N}{\log N}$, because n sending queues need to be identified and their state counter to be incremented. For $n > \frac{N}{\log N}$ it is more efficient to use $I = N$.

Table 5.3: $N \times N$ FC Problem Complexity C_{NN} for Stateless and Stateful FC Classes and the Required Amount of FC Information I.

Criteria	stateless	stateful
$I < N$	$N(C_{11} + U) + \mathrm{RXS}$	$NC_{11} + \mathrm{RXS}$
$I = N$	NC_{11}	NC_{11}

Here, complexity is found to be $C_{NN} = NC_{11}+\mathrm{RXS}$, because each sending buffer maintains its own accounting and decision unit. In case there is more information generated than can be drained, FC information will queue up and contend for reverse path access.

5.5.2 Examples

This problem also relates to both hop-by-hop as well as end-to-end FC.

For the hop-by-hop FC N sources are connected to N destinations, each one representing a source-destination pair, sharing a common link. This can be for instance the case for connecting multiple virtual circuits (communication area) [8], or virtual channels (multiprocessor area) over one link [33].

For end-to-end FC this is similar, however, there are N users, each one representing a source-destination pair, sharing networking resources.

5.5.3 Summary

The FC complexity C_{NN} together with the required amount of FC information I is summarized in Table 5.3. The minimum number of FC events that have to be returned in the next packet cycle is one for both stateless and stateful FC schemes. This and the performance related aspects and observations are similar to the $1 \times N$ combination. The $N \times N$ FC problem relates to the link-level for instance to realize multiple virtual channels over a link. But also, when viewing the network as shared medium, this problem also relates to end-to-end FC. The stateful FC schemes are found to be less complex than stateless schemes, in particular when the FC bandwidth is restricted. [3]

5.6 Switching

In this section we will map the N-dimensional FC problems as discussed in the previous subsections onto switching architectures relevant today. We will present the resulting complexity C_{xy} depending on FC information and on the class of FC, as introduced in Section 4.3.

5.6.1 Dedicated Input Port and Shared Output Queues

There is for instance the architecture of shared output queues with buffers at the input ports. An example of this architecture is the Vulcan switch and its successors [81] of the IBM's SP2 multiprocessor computer. Each input and output port implements a 1×1 flit-based relative update credit FC protocol at the link level. It requires a receive buffer at each input and a send buffer at each output port, which is shown in Figure 5.7. The send and receive buffer sizes are as large as to maintain a RTT worth of flits.

In general a switch built with this kind of FC mechanism at both the switch's ingress and egress conveys an amount of $I = 1$ of FC information per link for both stateless and stateful FC classes. The overall complexity is $C_{11,switch} = 2NC_{11}$, which happens to be independent of FC class. This is shown in Table 5.4, where the switch overall complexity is broken down into stateless and stateful schemes for both the switch's ingress and egress.

[3] If the FC bandwidth provides $I = N$, then the complexity difference is determined by C_{11} only, which itself is less complex for the selected stateful scheme.

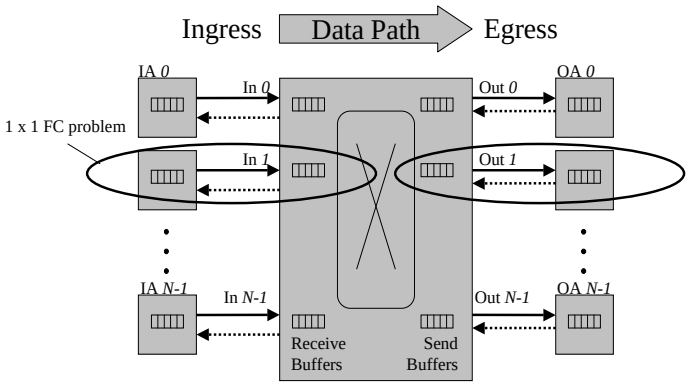


Figure 5.7: A Switch with 1×1 FC Problems

Table 5.4: Mapping of N-dimensional FC Problem onto Switching Architectures.

Criteria	Ingress		Egress	
	stateless	stateful	stateless	stateful
$I = 1$ $1 < I < N$ $I = N$	NC_{11}	NC_{11}	NC_{11}	NC_{11}

Clearly, speaking for FC complexity and bandwidth this architecture is very attractive, because it utilizes resources economically. However, from a performance point of view this architecture will suffer from the HOL-blocking phenomenon. This architecture seems therefore suitable for multi-processor environments, where the aggregate switch-bandwidth requirements are not as strict as for instance in high-speed switching in communications. As we strive for a converging architecture, this type of architecture is not further considered.

5.6.2 Combined Input and Output Queues (CIOQ) Switches

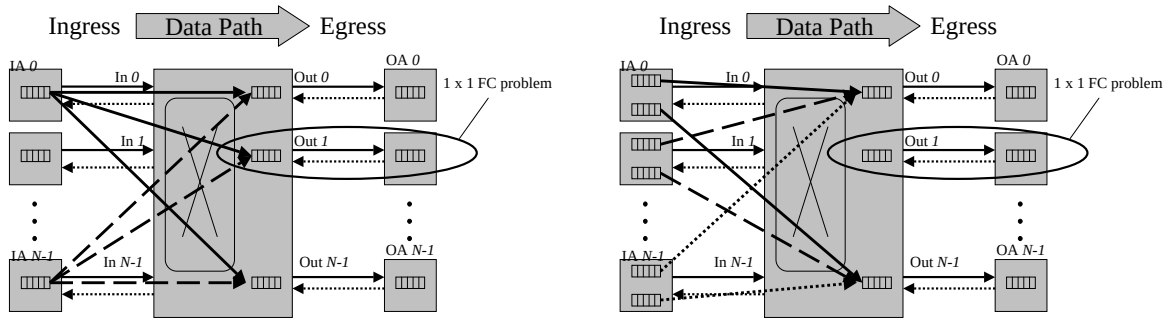


Figure 5.8: CIOQ Switching Architecture with 1×1 FC Problems at the Egress; a) CIOQ/FIFO: $1 \times N$ at the Ingress, b) CIOQ/VOQ: $N \times N$ at the Ingress

The FC problem at the ingress of CIOQ switching architectures depends on the input queueing strategy. When using FIFO, then there are N $1 \times N$ FC problem present. When using VOQs instead, then there are N $N \times N$ FC problems. In both cases there are N 1×1 FC problems at the egress. This is shown in Figures 5.8a) for CIOQ/FIFO and 5.8b) for CIOQ/VOQ switches. The ingress and egress FC problems map onto the link-level FC of Figure 5.1.

In CIOQ switches the output queues are shared which means that up to N sources may write to one output queue within one tu. This is independent of any input queueing strategy.

For *stateless* schemes this signifies that at any point in time each and every destination buffer must be able to stop its sources. Once a stop threshold by an arbitrary destination is reached, feasibility to this destination is withdrawn, i.e. a stop signal is broadcasted to all connected sources. The broadcast is a "must send" operation, i.e. it must happen in the next packet cycle. The FC bandwidth must provide enough bandwidth to cover the case that all N destinations reach the stop threshold at the same time. Then all destinations must stop all sources, which are all "must send" operations. Therefore, it is a requirement for the FC information to convey a vector of length N ($I = N$). Each position of this vector corresponds to a destination that possibly must be stopped. Reducing this vector by compression mechanisms is almost impossible, because the stopping may occur in any of 2^N combinations, which is traffic dependent, and hence not predictable. This FC vector, which is conveyed per FCD allows in turn to start and stop packet transmission for each FCSD.

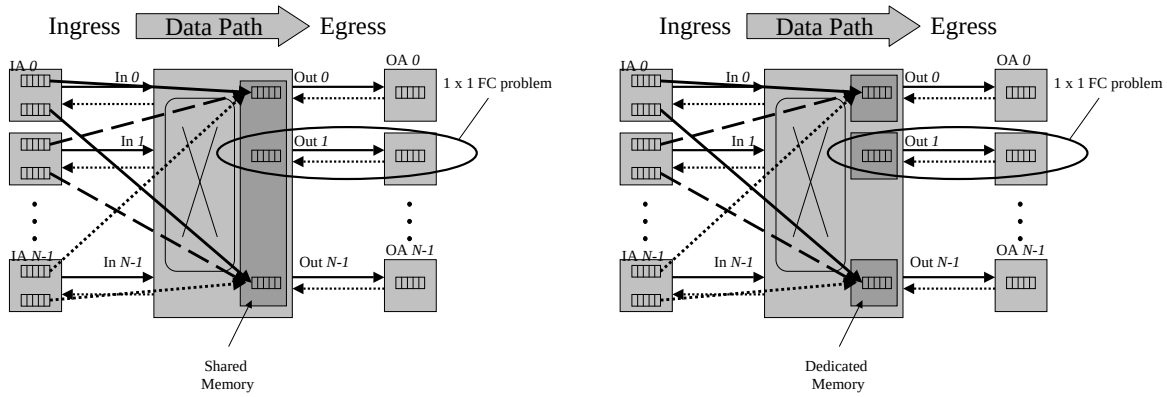


Figure 5.9: CIOQ/VOQ with; a) Shared, b) Dedicated Memory

For CIOQ switches with shared memory, where the total amount of memory locations is smaller than the offered buffer space across all output queues, two independent FC mechanisms coexist. One to protect against buffer overflow as described above, and one that protects against memory overflow. The latter is also a $N \times 1$ FC problem, because all N sources, i.e. IAs, have to be stopped, in case the memory reaches its critical threshold. This increases the amount of FC information to convey to $I = N + 1$. Such a scheme is for instance used in IBM's PRIZMA switch [37] and schematically shown in Fig. 5.9a) for CIOQ/VOQ.

For CIOQ switches using dedicated memories for each of the N output queues, the extra memory protection is not required. The buffer overflow protection prevents the memory from overflowing at the same time. This case is shown in Fig. 5.9b) for CIOQ/VOQ. The amount of FC information remains therefore at $I = N$.

The complexity of both the $1 \times N$ and $N \times N$ FC problems at the ingress of CIOQ switches is given by NC_{11}, because the N accounting and decision units at the receiving side (switch) are shared by all sources and there are only N thereof. At the egress this arrangement has N 1×1 FC problems with an aggregate complexity of also NC_{11}.

The feasibility information to be returned in *stateful* schemes are classified "can send". There can be up to N departures that originated all from the same source. In such cases there are up to N credits to be returned. As they are classified "can send", they do not have to be returned within one tu. The amount of FC information to return is $I = n \log N$, whereby n is a number depending on the available FC bandwidth. The output that made a departure needs to be encoded by $\log N$, because the IA must be able to associate the returned credit to the state counter of the related destination buffer. As the number of available packet locations at the reception side

is initially given as credits to the state counters at the transmission side, memory and buffer will never overflow. Therefore, the amount of FC information I to return and the resulting complexity are independent of the underlying memory structure of the switch. For the switch to be able to return the credits to IA where the packet originated, it must maintain N^2 state counters, i.e. N^2 FCSDs. This fact makes the complexity and amount of FC information I independent of the buffer structure.

A stateful scheme is for instance implemented in the ATLAS-switch [1] which uses a shared memory and N output queues. Here, multiple credits are queued for return (RXS) using a FIFO discipline and $n = 2$. It is not clear, whether n has an influence on performance or not. Clearly, as we have seen from Subsection 4.1.2 about performance of FC schemes, this will contribute to queueing time. In fact, since the number of credits to be returned to one IA per tu may be larger than one, different credits will contend for reverse path access.

The FC complexity is here at the ingress $NC_{1N} = N^2C_{11} + N\mathrm{RXS}$ for the same reasons that were given in Section 5.3. At the egress there are N 1×1 FC problems with a complexity of in total NC_{11}.

As a summary, the complexity and the optimized amount of FC information I for CIOQ switches are found to be using

stateless FC

- at the ingress:

$$\begin{aligned} C_{\mathrm{CIOQ,ingress}} &= NC_{11} \\ I_{\mathrm{CIOQ,ingress}} &= N^2 \end{aligned} \tag{5.3}$$

  because N accounting and decision units per output queue are shared by all inputs. The per-link FC information $I = N + 1$ for shared memory of size M_{tot}, whereby the number of locations in all output queues B_{OQ_i}, $\sum_{i=0}^{N-1} B_{\mathrm{OQ}_i} > M_{\mathrm{tot}}$. Otherwise $I = N$. Here, the amount of FC information to return must be at least $I = N$ per link as the idea of RXSs per input link can not be applied. Across all input FCDs, the amount of FC information is therefore in the best case N^2.

- at the egress:

$$\begin{aligned} C_{\mathrm{CIOQ,egress}} &= NC_{11} \\ I_{\mathrm{CIOQ,egress}} &= N \end{aligned} \tag{5.4}$$

  there are N 1×1 FC problems of complexity C_{11} each. Per 1×1 link the amount of FC information to convey is 1.

stateful FC

- at the ingress:

$$\begin{aligned} C_{\mathrm{CIOQ,ingress}} &= N^2C_{11} + N\mathrm{RXS} \\ I_{\mathrm{CIOQ,ingress}} &= N \log N, \end{aligned} \tag{5.5}$$

  because each IA maintains N accounting and decision units. The order of returning feasibility information is determined by the RXS (in case of a bandwidth limited FC channel). The per-link amount of FC information can be summarized as follows:

$$I = \begin{cases} n \log N & \text{if } n < \frac{N}{\log N} \\ N & \text{if } n \geq \frac{N}{\log N} \end{cases}$$

For the FC-bandwidth optimized case of $n = 1$, the amount of FC information per link is found to be $\log N$, hence for the entire ingress it is $N \log N$.

- at the egress:

$$\begin{aligned} C_{\text{CIOQ,egress}} &= NC_{11} \\ I_{\text{CIOQ,egress}} &= N, \end{aligned} \tag{5.6}$$

as there are N 1×1 FC problems of complexity C_{11} and $I = 1$.

The overall ingress and egress FC complexity of a CIOQ switch with FIFO or VOQ input queues has been summarized in Table 5.5 under particular consideration of the amount of FC information I to convey per input link, i.e. FCD. The table shows that a stateless FC scheme requires a FC channel capacity of $I = N$.

Table 5.5: Mapping of N-dimensional FC Problem onto CIOQ Switching Architectures.

Criteria	Ingress		Egress	
	stateless	stateful	stateless	stateful
$I = 1$	—	—	NC_{11}	NC_{11}
$\log N \leq I < N$	—	$N^2C_{11} + N\text{RXS}$	—	—
$I \geq N$	NC_{11}	N^2C_{11}	—	—

Assuming a FIFO queueing discipline, the $1 \times N$ FC problem will always suffer from the HOL blocking phenomenon. This is the reason, why it is not so interesting for switching and we will not further pursue it and therefore dismiss the FIFO input queueing organization from further analysis and focus on VOQ instead. CIOQ seems to be best suited for a stateful FC scheme at the ingress, if the goal is to reduce the amount of FC information to convey in particular if a relative credit update protocol is applied. An exact evaluation of complexity and FC information that is required for such an architecture for the case of "optimized" grants, as introduced in Subsection 4.3.1, and relative credit-update, as introduced in Subsection 4.3.3, will be performed in Subsection 5.6.4.

5.6.3 Combined Input and Crosspoint Queued (CICQ) Switches

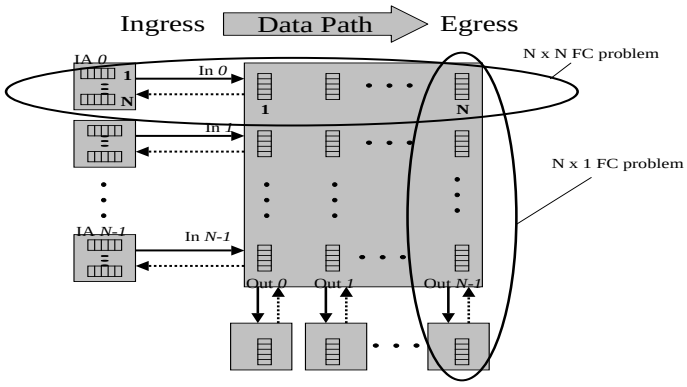


Figure 5.10: CICQ/VOQ Switching Architecture with $N \times N$ FC Problems at the Ingress and $N \times 1$ FC Problems at the Egress

The FC problem at the ingress of CICQ switches depends as in the case of CIOQ switches on the input queueing strategy. When using FIFOs, then there are N $1 \times N$ FC problems, for VOQs, there are N $N \times N$ such problems. The latter is shown in Figure 5.10. At the egress, and this is the difference from a FC perspective to CIOQ switches, there are – independent of the input queueing structure – N $N \times 1$ FC problems. These problems would then map to the link-level FC at the ingress side of Fig. 5.1.

For *stateless* schemes, we have basically the same discussion as in the CIOQ case (see Subsection 5.6.2). However, as there are N^2 output queues, each FCD has its own dedicated set of queues. This means that the amount of FC information I to return and complexity follow the reasoning of Subsection 5.5.1: At the cost of some extra complexity that is required for the RXS per FCD at the switch and state registers U at the sender, I can be reduced, if needed. Therefore, $C_{\mathrm{CICQ,ingress}} = NC_{NN} = N^2(C_{11} + U) + N\mathrm{RXS}$ and $I \leq N$. At the egress the complexity is found as $C_{\mathrm{CICQ,egress}} = NC_{N1} = NC_{11}$, because there is only one state and decision counter realized at the OAs.

For *stateful* schemes, and $N \times N$ FC problems per input link, the switch ingress FC complexity calculates according to Subsection 5.5.1 as $C_{\mathrm{CICQ,ingress}} = NC_{NN} = N^2(C_{11}) + N\mathrm{RXS}$. With the usage of an RXS per FCD, the amount of FC information generally is $I \leq N$. At the egress, complexity is $C_{\mathrm{CICQ,egress}} = NC_{N1} = N^2C_{11}$, because the state and decision units are located at the sender side (switch), where N queues try to access the output link. Hence each egress link has a complexity of NC_{11}.

To sum up, complexity and the optimized amount of FC information to convey is found in CICQ switches independently of the input queueing strategy (FIFO/VOQ) using:

stateless FC

- at the ingress:

$$\begin{aligned} C_{\mathrm{CICQ,ingress}} &= N^2(C_{11} + U) + N\mathrm{RXS} \\ I_{\mathrm{CICQ,ingress}} &= N \log N \end{aligned} \tag{5.7}$$

  For $I = N$ per FCD, the RXS unit was superfluous and therefore ingress complexity would reduce to N^2C_{11}. As we assume the presence of RXSs per FCD our goal of reducing the amount of FC information is met, because it reduces from $I = N^2$ down to $I = N \log N$.

- at the egress:

$$\begin{aligned} C_{\mathrm{CICQ,egress}} &= NC_{11} \\ I_{\mathrm{CICQ,egress}} &= N \end{aligned} \tag{5.8}$$

stateful FC

- at the ingress:

$$\begin{aligned} C_{\mathrm{CICQ,ingress}} &= N^2C_{11} + N\mathrm{RXS} \\ I_{\mathrm{CICQ,ingress}} &= N \log N \end{aligned} \tag{5.9}$$

  There are $N \times N$ FC problems per link that each require an RXS and a complexity of NC_{11}. In the costly case of $I = N$, the RXS unit is not required, reducing the

complexity to N^2C_{11}. The requirements of the amount of FC information can be summarized as

$$I = \begin{cases} n \log N & \text{if } n < \frac{N}{\log N} \\ N & \text{if } n \geq \frac{N}{\log N} \end{cases}$$

Since we strive for the lowest FC bandwidth utilization, we select $n = 1$ and find the results of Eq. 5.9 for all N ingress links.

- at the egress:

$$\begin{aligned} C_{\text{CICQ,egress}} &= N^2 C_{11} \\ I_{\text{CICQ,egress}} &= N \log N \end{aligned} \tag{5.10}$$

There is a $N \times 1$ FC problem per output, which equals a complexity of NC_{11}. The amount of FC information is here found to be $\log N$ per link as the individual sources have to be identifiable.

The complexity is summarized in Table 5.6 for CICQ switches according to the amount of FC information required.

Table 5.6: Mapping of N-dimensional FC Problem onto CICQ Switching Architectures.

Criteria	Ingress		Egress	
	stateless	stateful	stateless	stateful
$I = 1$	—	—	NC_{11}	—
$\log N \leq I < N$	$N^2(C_{11} + U) + N\text{RXS}$	$N^2C_{11} + N\text{RXS}$	—	N^2C_{11}
$I = N$	N^2C_{11}	N^2C_{11}	—	—

5.6.4 Application Example: Optimized Grant and Relative Credit

In this Section we will use the results of the last two subsections to compare the complexity and required FC bandwidth of CIOQ and CICQ switches for stateless and stateful FC schemes. A RXS is used, where applicable. FC scheduling per FCD by the RXS module allows to reduce the amount of FC information to convey from $n = N$ down to one. Thus, we assume $n = 1$, where possible.

In order to be able to carry out this comparison, we have to use the results of Table 4.1 describing the base complexity C_{11} for stateless and stateful FC protocols and apply them onto the complexity found for CIOQ and CICQ switching architectures as described in Table 5.5 and 5.6. We choose those candidates C_{11} that had the lowest complexity of its class. For better distinction of the complexity, we introduce the upper index L and F, for state*l*ess and state*f*ul complexity C_{11}^L and C_{11}^F, respectively. For stateless FC, we choose the *optimized grant*, which has a complexity of

$$C_{11}^L = 3. \tag{5.11}$$

For stateful FC we select the *relative credit* with a complexity of

$$C_{11}^F = 2. \tag{5.12}$$

So, if we talk about stateless and stateful FC schemes in the following, we have these two implementations in mind. Furthermore, we assign the complexity for the RXS to be $N + 1$,

assuming that each RXS requires N accounting units and also one decision unit. We already considered both unit types to be of equal complexity for evaluating stateless and stateful FC schemes. The unit U is assumed to be of complexity 1.

For the calculation of the minimum aggregate FC bandwidth, we have to multiply the minimum amount of FC information I per FCD with the switch dimension N. We then have to add the result of ingress and egress, and finally multiply the result with the ratio Y/P, whereby Y signifies the port speed in bits/s and P is the assumed packet length in bytes. For our calculations we assume $P = 64$ bytes and $Y = 10$Gb/s (OC-192).

CIOQ switches

We have four possible combinations of using stateless and stateful FC at ingress and egress. We list these FC class pairs in the following and show how their total complexity C_{tot} as a function of switch size N is found. Additionally, we calculate the aggregate FC bandwidth $BW_{\text{FC,agg}}$. The following list is headed by a pair of FC classes, whereas the first item describes the FC class utilized at the ingress, and the second element the one associated with the egress. Indices not specifically introduced are self-explanatory. We use Eqs. 5.11 and 5.12 where applicable.

a) stateless/stateless: Using Eq. 5.3, and 5.4 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^L_{\text{CIOQ,ingress}} + C^L_{\text{CIOQ,egress}} &&= 2NC^L_{11} &&= \\ &&&= 6N; \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^L_{\text{CIOQ,ingress}} + I^L_{\text{CIOQ,egress}}) &&= \tfrac{Y}{P}(N^2+N) \end{aligned}$$

b) stateless/stateful: Using Eq. 5.3, and 5.6 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^L_{\text{CIOQ,ingress}} + C^F_{\text{CIOQ,egress}} &&= N(C^L_{11} + C^F_{11}) &&= \\ &&&= 5N; \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^L_{\text{CIOQ,ingress}} + I^F_{\text{CIOQ,egress}}) &&= \tfrac{Y}{P}(N^2+N) \end{aligned}$$

c) stateful/stateless: Using Eq. 5.5, and 5.4 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^F_{\text{CIOQ,ingress}} + C^L_{\text{CIOQ,egress}} &&= N^2(C^F_{11}+1) + N(C^L_{11}+1) &&= \\ &&&= 3N^2 + 4N; \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^F_{\text{CIOQ,ingress}} + I^L_{\text{CIOQ,egress}}) &&= \tfrac{Y}{P}N(\log N + 1) \end{aligned}$$

d) stateful/stateful: Using Eq. 5.5, and 5.6 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^F_{\text{CIOQ,ingress}} + C^F_{\text{CIOQ,egress}} &&= N^2(C^F_{11}+1) + NC^F_{11} &&= \\ &&&= 3N^2 + 3N; \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^F_{\text{CIOQ,ingress}} + I^F_{\text{CIOQ,egress}}) &&= \tfrac{Y}{P}N(\log N + 1) \end{aligned}$$

We can split the results in two groups. One group that shows a low FC complexity, but high aggregate FC bandwidth, and another group that exhibits the inverse behavior, a high FC complexity and a low required FC bandwidth. This is shown in Table 5.7 which shows both the FC complexity and FC bandwidth for switch sizes of $N = 8, 16, 32,$ and 64. Cases a) and

Table 5.7: Total Complexity and Aggregate FC Bandwidth in Gb/s for CIOQ Switches.

N	a)		b)		c)		d)	
	C	FC BW	C	FC BW	C	FC BW	C	FC BW
8	48	1.40625	40	1.40625	224	0.625	216	0.625
16	96	5.3125	80	5.3125	832	1.5625	816	1.5625
32	192	20.625	160	20.625	3200	3.75	3168	3.75
64	384	81.25	320	81.25	12544	8.75	12480	8.75

b) with a stateless FC protocol at the ingress belong to the first group. In both cases the FC bandwidth is calculated up to 81.25 Gb/s for a 64×64 SF. A stateful FC protocol at the egress shows a slightly lower FC complexity than a stateless protocol. Cases c) and d) with a stateful FC protocol at the ingress belong to the other group that requires roughly 1/10th of the FC bandwidth, but a much higher complexity. In particular, a 64×64 SF needs in the case of a stateless FC protocol at the egress 12,544 logical units on the switch, whereas with a stateful protocol, slightly less namely 12,480 such units are needed. We can say that the required FC bandwidth depends on the ingress FC class. *The large savings in FC bandwidth for a stateful scheme at the ingress is due to the fact that here the RXS can be applied, which is not true for stateless schemes*. Overall case d) with a stateful FC class at both the ingress and egress seems to be the most attractive for CIOQ switches as it shows low FC bandwidth overhead at relatively low complexity.

CICQ switches

We do the same calculation for CICQ switches and list the results according to the previous example using CIOQ switches and the complexity for the specific stateless and stateful schemes expressed in Eqs 5.11 and 5.12.

a) stateless/stateless: Using Eq. 5.7, and 5.8 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^L_{\text{CICQ,ingress}} + C^L_{\text{CICQ,egress}} &&= N^2(C^L_{11}+1) + N(N+1) + NC^L_{11} = \\ &&&= 5N^2 + 4N \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^L_{\text{CICQ,ingress}} + I^L_{\text{CICQ,egress}}) &&= \tfrac{Y}{P}N(\log N + 1) \end{aligned}$$

b) stateless/stateful: Using Eq. 5.7, and 5.10 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^L_{\text{CICQ,ingress}} + C^F_{\text{CICQ,egress}} &&= N^2(C^L_{11}+1) + N(N+1) + N^2C^F_{11} = \\ &&&= 7N^2 + N \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^L_{\text{CICQ,ingress}} + I^F_{\text{CICQ,egress}}) &&= \tfrac{Y}{P}2N\log N \end{aligned}$$

c) stateful/stateless: Using Eq. 5.9, and 5.8 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^F_{\text{CICQ,ingress}} + C^L_{\text{CICQ,egress}} &&= N^2(C^F_{11}+1) + N(C^L_{11}+1) = \\ &&&= 3N^2 + 4N \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P}N(I^F_{\text{CICQ,ingress}} + I^L_{\text{CICQ,egress}}) &&= \tfrac{Y}{P}N(\log N + 1) \end{aligned}$$

d) stateful/stateful: Using Eq. 5.9, and 5.10 the resulting total FC complexity C_{tot} and aggregate FC bandwidth $BW_{\text{FC,agg}}$ yields

$$\begin{aligned} C_{\text{tot}} &= C^F_{\text{CICQ,ingress}} + C^F_{\text{CICQ,egress}} &&= N^2(C^F_{11}+1) + N^2 C^F_{11} + N = \\ &&&= 5N^2 + N \\ BW_{\text{FC,agg}} &= \tfrac{Y}{P} N (I^F_{\text{CICQ,ingress}} + I^F_{\text{CICQ,egress}}) &&= \tfrac{Y}{P} 2N \log N \end{aligned}$$

Table 5.8: Total Complexity and Aggregate FC Bandwidth in Gb/s for CICQ Switches.

N	a)		b)		c)		d)	
	C	FC BW	C	FC BW	C	FC BW	C	FC BW
8	352	0.625	456	0.9375	224	0.625	328	0.9375
16	1344	1.5625	1808	2.5	832	1.5625	1296	2.5
32	5248	3.75	7200	6.25	3200	3.75	5152	6.25
64	20796	8.75	28736	15	12544	8.75	20544	15

The results here can also be partitioned into two groups. One group shows low, the other one high FC bandwidth. This is shown in Table 5.8 for four SF sizes: $N = 8, 16, 32,$ and 64. Cases a) and c) with a stateless FC scheme at the egress belong to the first group requiring low and equal FC bandwidth. For instance, a 64×64 device needed 8.75 Gb/s aggregate FC bandwidth in both cases. A comparison within this group reveals that the total FC complexity is lower given that a stateful scheme at the ingress is used. As an example, for a 64×64 device total complexity is reduced from roughly 21,000 logical units down to about 12,500.

The other group is identified by cases b) and d) with a stateful scheme at the egress. Here, the FC bandwidth is about twice as large, namely 15 Gb/s. Comparable to the first group, a stateless scheme at the ingress would be more costly in terms of total complexity than a stateful scheme.

By nature of its dedicated buffers per FCSD, a CICQ switch does allow the usage of the RXS independent of the underlying FC class. The FC bandwidth is thus dominated by the egress FCDs. Here, it is more efficient to use a stateless scheme. Overall, it seems that the combination of a stateful scheme at the ingress, but a stateless protocol at the egress is the most promising in terms of FC bandwidth efficiency and total FC complexity.

5.7 Conclusions

We have identified the levels of FC that exist in networking nodes, or in multiprocessor networks respectively. We presented examples for each level. We are therefore able to exactly understand, differentiate and locate the FC problem that we are addressing in this thesis. We are addressing the link-level FC that interacts between switch and its surrounding adapters.

We have discussed the different FC arrangements and presented their complexity, as well as their required amount of FC information to convey. We have thereby introduced an developed the concept of FCDs. We applied this concept onto CIOQ and CICQ switch architectures to evaluate FC complexity and aggregate FC bandwidth in the light of both stateless and stateful FC. For CIOQ SFs a stateless scheme at the ingress is attractive from a complexity point of view. However, the required FC bandwidth is not supportable. Owing to lower required FC bandwidth we consider a stateful scheme using a relative credit update at ingress and egress as more favorable according to Table 5.7. For CICQ SFs, a stateful scheme at the ingress and a

stateless scheme at the egress is less complex according to Table 5.8. We have further found that from a FC complexity point of view, the input queueing strategy (FIFO/VOQ) does not play a role.

We have found that generally there is no requirement to increase the FC bandwidth to more than 1 FC event, when scaling the number of buffers at either side of the common link. In order to be able to run the SF with a FC bandwidth of 1 FC event, we have introduced a device, the RXS, that allows to serialize and prioritize the return of feasibility information. The minimum amount of FC information that is associated to a FC event for a stateful FC scheme is $I = \log N$.

We have further identified the $N \times 1$ FC problem to be related to data path scheduling. Since such schedulers are built to provide QoS guarantees, it has to be considered in particular when dealing with the output side of a CICQ SF.

We have seen at the example of the $1 \times N$ FC problem that the order of return of feasibility information has an impact on the performance of the whole system.

We have demonstrated that the CICQ switch architecture is composed of N independent $N \times N$ FC problems at the ingress side of the SF and $N \times 1$ data path scheduling problems at the egress. This architecture and a stateful scheme seems therefore to be very attractive for further analysis, because – besides the considerations of Section 2.1.4 – it allows to analyze and optimize the packet arrival and departure process per FCD independent of the other FCDs. Moreover, an RXS module can be used independent of the underlying class of FC, which will let our results of Chapter 7 appear in a broader perspective. As we believe, the packet arrival process can be influenced by an RXS module. Once the packets are stored in the switch, i.e. they are buffered at each output per input, a data path scheduler can be used to provide QoS guarantees, which makes it attractive for communication switches. But it is also useful for multi-processor switches, as this architecture conceptually has the flexibility to implement any output (data path) scheduling algorithm without having to rearrange the order of packets queued.

These are the reasons that we believe justify the selection of a CICQ switch and a relative credit-update protocol at the ingress for further analysis. The believes and argumentation around FC scheduling and the RXS, as well as scheduling the output, we want to investigate in Chap. 7 by means of simulations using the unified model presented in Section. 2.3.

Having performed the analysis up to now gives us the foundation to understand the contribution of this thesis. We want to elaborate on it in the next chapter.

Chapter 6

Switch Architecture Design using Flow Control Domains (FCDs)

Link-level FC in interconnection networks operates between every transmitter-receiver pair. In single-stage buffered SFs it is active between switch and adapters, in multi-stage fabrics also in-between stages. Its primary function is to provide correctness, i.e. to prevent buffer overflow as pointed out in Section 4.1. Providing correctness avoids performance degradation in times when output links are oversubscribed [23]. FC is typically realized either by stateless schemes (Subsection 4.3.1), where the reception side reacts upon a buffer state change that, for example, causes a threshold overstep and stops packet transmission, or by stateful schemes (Subsections 4.3.2, 4.3.3), where the sender on its own behalf stops transmitting further packets to avoid buffer overflow. Grant FC falls into the category of stateless schemes, whereas credit FC is representative for stateful protocols. In buffered switching architectures, FC is executed at the ingress per input link and at the egress per output link.

We have seen that buffers at both sides of a link generally span an independent entity, an FCD. Considering a CICQ switch architecture with a VOQ input queueing organization, we identified N such FCDs of dimension $N \times N$ at the ingress and N FCDs of dimension $N \times 1$ at the egress. We conceive the switch as an aggregation of FCDs. This approach allows us to decompose the switch into a multiplicity of individual FC problems, which we can investigate separately. We have done this extensively in the two preceding chapters. We will now re-assemble the results and propose that future switch design and design of interconnection networks be led by *autonomous FCDs* and *FC scheduling*.

Autonomous FCDs

The concept of FCDs derived in Chap. 5 allows us to isolate a link together with its send and receive buffers from any kind of interconnection network. Typically, in interconnection networks multiple independent flows share a common link, which justifies multiple send and receive buffers. A FCD in isolation is shown in Fig. 6.1. The figure is split into a data path going from left to right in the upper part and a control path from right to left in the lower part. While the data path scheduler (DPS in the figure) schedules packets, the RXS schedules FC events. FC events are generated upon packet departures, which in turn are stored until they are served by the RXS. Several service disciplines will be introduced in Section 7.5. The amount of FC information that returns to the sender determines the required FC bandwidth and provides feasibility information to the data path scheduler.

The idea is that such an arrangement is self-governing: FC depends solely on local state information. If a reception buffer is part of a larger buffer that is for instance shared with

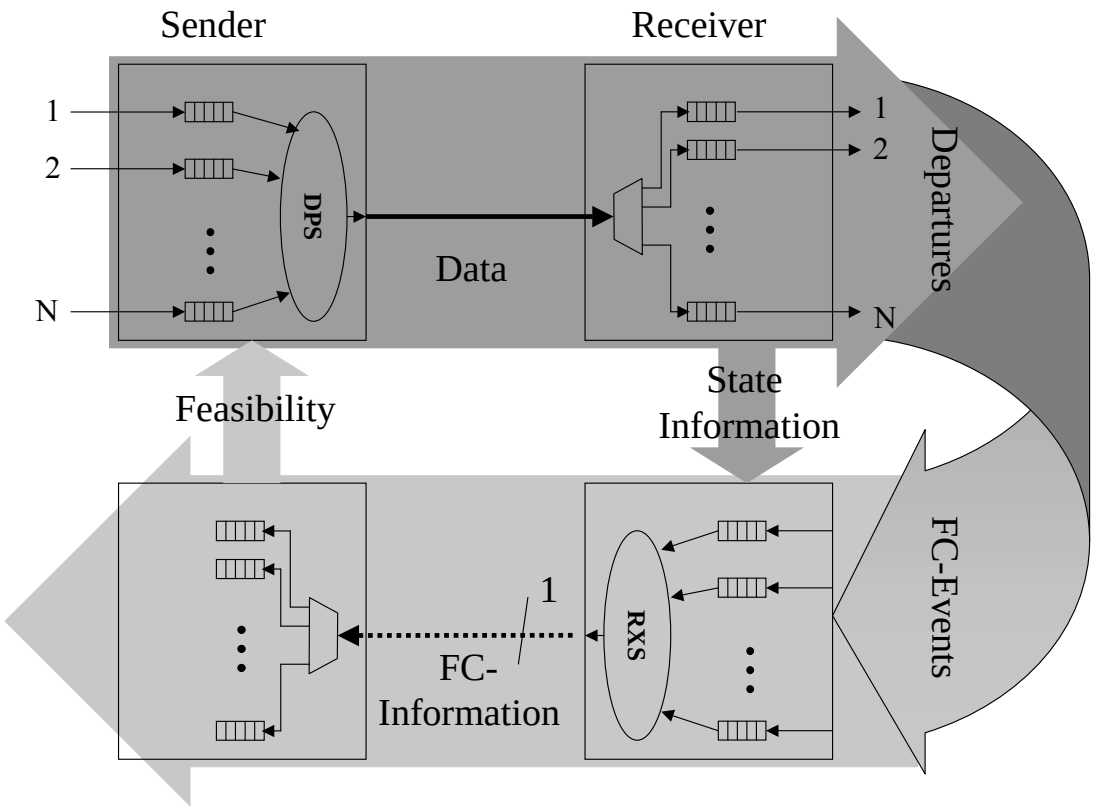


Figure 6.1: Flow Control Scheduling of a Single Link

connections of other FCDs, then only local, and not state information of the whole buffer, i.e. global state information is relevant for the system to be autonomous. In general, buffers not directly associated with the FCD under study are not considered for the generation of FC information. This makes the exchange of state information across FCDs, typically found in networking nodes superfluous.

Applied to switch design this approach has several advantages. Each FCD of the N input and output links operates autonomously as if the others were no present. Such a system is efficient, because it allows the amount of FC information to convey to be kept at a minimum as the state information of other FCDs is not required and consequently does not have to be conveyed. It is also scalable at relatively low cost, as FCDs can be added or removed without impacting performance or the communication overhead of existing links. With autonomous FCDs a switch is conceived as an aggregation of FCDs as shown in Fig. 6.2, whereas each connecting pair is detailed according to Fig. 6.1; the data path is shown as a solid line, whereas the control path is dotted. If the physical limit of such an aggregation is reached, it could accordingly be split and realized in two or more such aggregations that themselves are connected by FCDs. In this way multi-stage SFs may be realized. Given an appropriate algorithm, the efficiency of such an autonomous system is accomplished by FC scheduling, which we will explain in the following.

FC Scheduling

FC scheduling is the proposed mechanism to circumvent the interconnection bottleneck mentioned in Subsection 2.1.4. We take a single link to explain the behavior of FC scheduling in the context of switching. It is applied to all links that go to or from a switch and is not limited to this particular application, but can be deployed between switches and SFs as well as between networking nodes. FC scheduling is implemented through the RXS, which was also employed

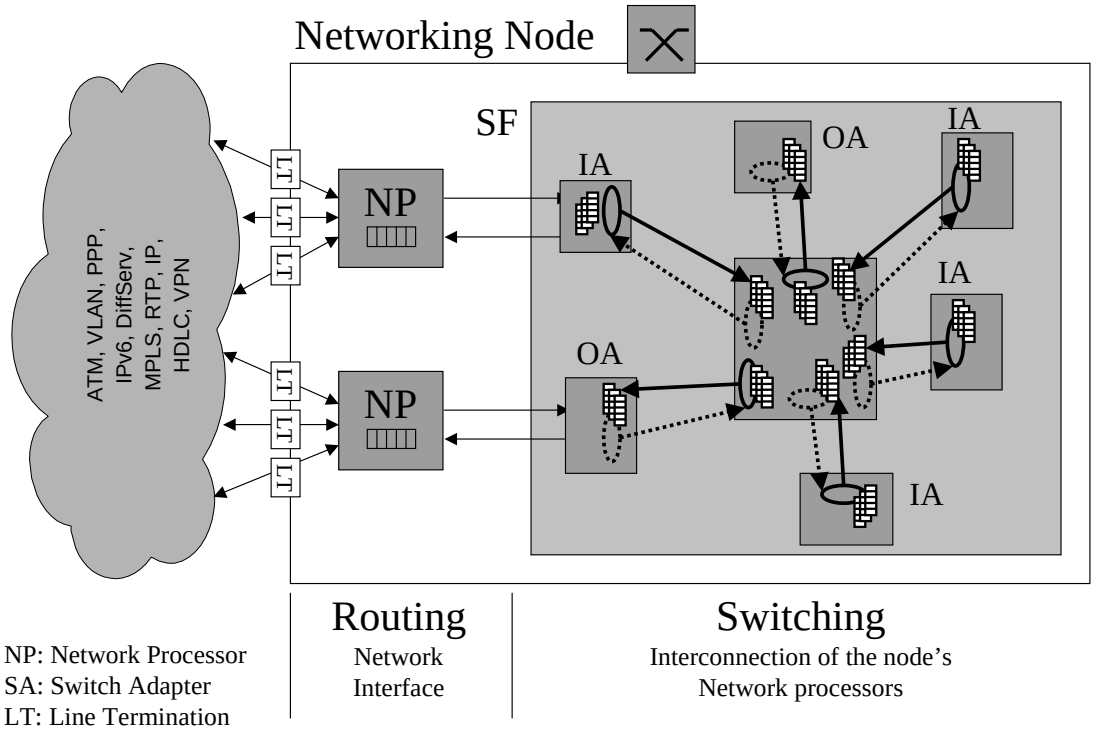


Figure 6.2: Switch Fabric Composed of Autonomous Flow Control Domains

in Figure 6.1.

The reception side of an ingress FCD is characterized by at most one packet arrival, but up to N concurrent departures per packet cycle, which has been shown in Chapter 5. These are explained by temporal output contention, where it may happen that all N or a subset of all output queue schedulers select packets for transmission that originate from the same switch row, i.e. FCD, and will be quantified in Subsection 7.3.2. As all output queue schedulers work independently of each other, these parallel departures are of random nature. This raises the question of bandwidth provisioning of the FC channel. Providing bandwidth to convey N FC events is sufficient for the worst case of returning N FC events; all the state information of the reception side is returned to the sending node. Thus, the data path scheduler of the sender collects all information, before making its decision to dispatch a new packet. From the perspective of a single autonomous FCD, this is a centralized approach.

Motivated by the discussion of Subsection 5.6.4, we assume a relative credit update FC protocol. The simplest encoding of a credit requires $\log N$ bits. The amount of FC information I to convey per packet cycle and per FCD may thus generally be calculated as $I = n \log N$, whereby $1 \leq n \leq \frac{N}{\log N}$. The minimum capacity required is $n = 1$ in order to maintain an equilibrium of packet arrivals and credit returns.

Reducing the FC channel bandwidth to its minimum of $I = \log N$ bits per packet cycle is desirable for system-level efficiency and scalability. However, this reduction evokes temporal contention of credits. Credit contention during phases of parallel departures leads to an increased backlog of credits. If not properly addressed, a data path scheduler may have insufficient credits available to maintain 100% upstream link utilization. This may adversely affect performance, and is a typical drawback of the bandwidth-constraint FC channel. On the other hand, owing to credit contention, FC scheduling also bears the potential to optimize performance. This is explained by means of two scenarios.

First, we assume a situation where credits contend for access to the reverse path. Credits are stored in FIFO order, which means that credits generated earlier also return earlier to their

corresponding VOQ. In this example, a credit at the head of the FIFO happens to be returned to a VOQ that has no packets to send. Meanwhile, a credit of another FCSD further back in the credit FIFO is needed by its corresponding VOQ to render an eligible packet feasible. The data path scheduler decides which packet to dispatch according to eligibility and feasibility. In the former case, no packet will be selected by the data path scheduler in the current packet cycle, because the VOQ that has a packet available does not have the feasibility information, and the empty VOQ has received the feasibility information, but does not have any packets to send. As a consequence, the system is not work-conserving. If the order of returning these two credits could have been inverted, this time slot could have been used. From this simple scenario we conclude that the order of returning FC information, i.e. FC scheduling, has the potential to influence performance.

Second, we assume a similar situation as just described with the addition that the cross-point memory associated to the credit at the head of the FIFO is almost full as are other cross-points in the same switch column. The cross-point memory of another FCSD associated with a credit further back in the credit FIFO is empty as are all other cross-points in this switch column. We assume that the two VOQs involved have packets that are eligible, but await feasibility. Without prioritizing the return of credits, the next packet will be sent to a cross-point that is already filled, contributing to even more output contention. However, with FC scheduling and scheduling said credit further back first, the other packet would be dispatched and could make immediate forward progress at the switch level, contributing to an improved performance. From this simplified scenario we conclude that the ordering of returning FC information improves performance.

Therefore, we think of FC not only as a means to assure correctness, but also as a means to enhance performance.

We now take the two scenarios and compare the consequences of FC-scheduling with an unscheduled FC channel of bandwidth N. For the first scenario and FC scheduling, the information of VOQ emptiness is required at the reception side. Otherwise it is impossible to prioritize a credit for a non-empty VOQ. This will cost additional overhead which is on the order of $\log N$, see Section 7.5. For an FC bandwidth of N, this would not be an issue.

For the second scenario and FC-scheduling, the cross-point memory occupancy is needed, to make an appropriate selection for credit return. For an FC bandwidth of N, the occupancy level of the transmit-side credit-pools can be used to achieve the same effect. This in turn will make the data path scheduler more complex. In FC scheduling this decision is distributed: The process of taking the credit-pools into account is already taken care of by credit prioritization through the RXS scheduler, and the data path selection process can be left to a RR.

For a FC bandwidth of N the same results are achieved as with FC scheduling, at the cost of a more complex data path scheduler and at the expense of FC bandwidth. It is important to us that

FC scheduling enables the distribution of data path scheduling across a link for all FCSDs of a given FCD.

This means that we have to expand the family of CIOQ switches by a new member. While the bufferless CIOQ architecture applies a centralized scheduling approach that is central with respect to all FCDs, the buffered CIOQ architectures distributes scheduling between FCDs (inter FCD). Our new approach allows scheduling to be distributed within a FCD (intra FCD).

In the following, we want to compare the aggregate FC overhead for all three subclasses of CIOQ switches to demonstrate the value of our result. For comparison we introduce the

notation "D/SD CIOQ switch", which describes whether a CIOQ switch applies scheduling that is central or distributed between all FCDs and within all FCDs, i.e. between all FCSDs of an FCD. D stands for FCD, SD for FCSD. Both can take the values *c*entralized or *d*istributed. For example, the CIOQ crossbar therefore is a c/c CIOQ switch, because it is central with respect to all FCDs and FCSDs. Accordingly, the new subclass is a d/d CIOQ switch. We assume FC scheduling only to be implemented in our proposed d/d architecture.

c/c CIOQ switch

A cross-bar CIOQ needs to send all FC information to a scheduler that determines which VOQs are allowed to transmit a packet in the next cycle. Typically, all eligible VOQs send a *request* to the centralized scheduler. The scheduler makes a decision and finally selects the appropriate VOQs via a *grant* (selection) signal. It also configures the cross-bar accordingly, which is done by a *configuration* signal to the cross-bar. If an output queue is blocked, a stop signal is created for the scheduler to stop selecting packets to blocked outputs. Therefore, the information flowing between VOQs, switch and centralized scheduler is quantified as follows. There are N^2 requests and N stop signals. Furthermore, there are N grant and N configurations signals. While the requests and stops have a capacity of 1, the grant and configuration have a width of $\log N$ bits. The bandwidth is calculated using the port speed Y and the length of a packet P. The total amount of control bandwidth therefore is $BW_{\text{tot}} = \frac{Y}{P}(N^2 + 2N \log N + N)$ without considering any possible speed-up.

In such a switching arrangement, the centralized scheduler has full knowledge of the state of the entire switch. Without obeying time constraints, the centralized scheduler is able to find an optimal scheduling decision. It determines which VOQs are allowed to send in the next packet cycle, hence traffic is being *pulled*. Such scheduling processes are complex to realize, and sometimes infeasible.

d/c CIOQ switch

A buffered switch splits the decision-making process across all N IA. Each adapter makes its own decision, independently. The amount of FC bandwidth needed between adapters and switch is calculated as follows. The switch returns FC information I of width N to each adapter per tu. Additionally, each OA sends a stop signal to the switch, if its buffer is full. Thus, the total amount of control bandwidth required calculates as $BW_{\text{tot}} = \frac{Y}{P}(N^2+N)$. In comparison to the bufferless switch, it means a reduction of $\frac{Y}{P}2N \log N$, because the selection and configuration signals are no longer required.

In such a switching arrangement, the distributed schedulers have only limited knowledge of the state of the entire switch. In fact, they assume the switch always to be able to accept further packets until notification from the switch to stop sending. The schedulers per IA are therefore able to *push* traffic into the switch until FC kicks in to assure correctness.

d/d CIOQ switch

A buffered switch that falls into this category subdivides the scheduling process not only across, but also within FCDs. Distribution of scheduling within a FCD means that not only the sending side of a FCD decides which packet to transmit next, but also the reception side participates in this process.

The amount of FC bandwidth needed between adapters and switch calculates as follows. The amount of FC information to return per ingress FCD needs only be $I = \log N$ bits, whereas each egress FCD requires $I = 1$. Therefore, the total amount of FC bandwidth is $BW_{\text{tot}} =$

$\frac{P}{L}(N \log N + N)$. Compared with the c/c and d/c CIOQ switch, the control bandwidth scales with $N \log N$, instead of N^2, which translates into considerable savings.

This scheme can be considered a hybrid of the previous two alternatives. This is, because under low loads, the switch operates as a d/c switch. Traffic is pushed into the switch. The system operates as if the switch was always able of accepting further packets. The state of the switch is not of interest. However, if the load is increased, i.e. under high loads, the switch begins to work as a c/c switch. Under such circumstances, the switch buffers start to fill more and more, as there will be more contention situations. The state of the reception, i.e. switch buffers, is taken into account in the scheduling decision, which is equivalent to a c/c switch. Although the buffer state is being given implicitly by the order of the return of FC information, the effect is the same: the switch pulls traffic from the adapters. This similarity to the c/c switch is accompanied by a desirable reduction in FC overhead.

Comparison

Full knowledge of the entire switch (c/c CIOQ switch) comes at the cost of an increase in FC overhead, switch cost and scheduler complexity. The aggregate switch FC bandwidth overhead (BW_{tot}) is quantified in Table 6.1 for switch sizes of $N = 8, 16, 32,$ and 64 using a packet length of $P = 64$ Bytes and a port speed of $Y = 16$ Gb/s. The values are given in Gb/s.

The distributed scheduling approach of a d/c switch already achieves some savings. The savings decrease as the switch dimension increases: 40% for an 8-port device, but only 16% for a 64-port switch.

The amount of FC bandwidth is reduced even more drastically for a d/d switch architecture. We therefore refer to this solution as being scalable. Here, the savings in FC bandwidth are up to 90% for the switch sizes given in Table 6.1. The savings increase as the switch dimension increases. For an 8-port switch, the savings are 71%, whereas for a dimension of 64 the savings are 91% with respect to a c/c architecture.

Table 6.1: Quantification of the Total FC Overhead in Gb/s for Various Subclasses of CIOQ Architectures and System Sizes N.

N	c/c	d/c	d/d
8	3.75	2.25	1
16	12.5	8.5	2.5
32	43	33	6
64	154	130	14

In the next chapter, we will evaluate the performance of a d/d switching architecture under various traffic assumptions, system parameters, and RXS-scheduling strategies and proof by means of performance simulation that the concept of autonomous FCDs and FC scheduling is practical.

Chapter 7

The Reception Scheduler

From our objectives stated in Chapter 3, we have achieved so far a classification of FC schemes and a quantification of FC complexity, information and resource consumption – for a single link in Chapter 4 and for multiple buffers sharing a common link and various switch architectures in Chapter 5. On this basis and together with the concept of autonomous FCDs and FC scheduling proposed in Chapter 6, we described a d/d CIOQ switch that offers good scalability at low FC overhead. A d/d CIOQ is a compromise between two diverging switch architectures: one that uses full state information at the ingress schedulers and one that operates without state information. The compromise requires to schedule FC events, which is the function of the *reception scheduler* (RXS).

In this chapter, we want to proof that this compromise is a valid approach by means of performance simulations. We will thereby obey our objective of optimizing FC overhead to achieve better bandwidth utilization and especially consider fabrics with large RTTs. We will present and discuss various RXS algorithms.

The analysis of this chapter concerns mainly the ingress side of a SF, but is also applicable to the egress, in particular in the case of multi-stage fabrics. Furthermore, we want to provide insight into the necessities of output-queue scheduling, which is related to the SF's egress.

The key questions related to the RXS and output queue scheduler (OQS) will be presented in Section 7.1. After having outlined these key questions and discussed credit scheduling and the RXS in Sections 7.3 to 7.5, we will disclose the performance results that we obtained by means of simulations in Section 7.6. For the simulations we use 3 system parameters, switch size N, link length $\mathrm{RTT}/2$, and memory size M with 36 levels in total. For clarity reasons, the system parameters used are summarized in a table per experiment. Both simulation technique and unified model were already presented in Chap. 2. As motivated in 5.7, we will use a CICQ switch architecture for our simulation effort and employ a relative credit-update FC protocol between IA and switch. Before we can analyze a flow control path scheduler, we have to determine and give reason to the selection of the data-path schedulers. This is done in Section 7.2 which is considered a prerequisite to our performance evaluation.

7.1 Key Questions

7.1.1 Ingress - $N \times N$ FC problem

The questions that we will pursue concerning the switch's ingress, i.e. the RXS, are related to:

1. RXS potential;

2. RXS strategy;

3. RXS impact;

We want to address the following key issues, using the evaluation methodology that was presented in Sec. 2.4.

- Does the propagation time have an impact on the overall switch performance? By keeping the single flows within a switch work-conserving, what is the influence of varying link-length?

- What is the impact of queueing time? Specifically, we want to study the impact of a bandwidth constrained reverse channel employing a RXS. To this end, we will analyze two extreme alternatives. One that offers an unscheduled reverse-path of bandwidth N, referred to as our *Reference*. Here, an RXS is superfluous, because the reverse-path bandwidth is unconstrained. Another one that offers a scheduled reverse-path, which reduces this bandwidth to only one control information per tu. While the unscheduled case will exclude queueing time, the scheduled case will induce a certain queueing time and a phenomenon referred to as "credit contention". Credit contention in the latter case may only be observed in the presence of more than one departure. What are reasonable assumptions about the occurrence of such "parallel departures"? Questions worth asking are: What is the potential that credit contention bears, what is the danger? What are reasonable RXS strategies? Which RXS strategy exploits credit contention best? Are these strategies robust against changing traffic conditions?

- What is the impact of correctness time on switch performance? In case of long links, e.g. links of RTT ≥ 10 the total amount of memory seems to be very large. If the switch should also support different classes of traffic, then even more memory will be required. Therefore, we are interested in the minimum amount of memory that required for a single class of traffic, such that for a given amount of memory, which in turn is mostly given by technology, multiple classes of traffic can be supported.

7.1.2 Egress - $N \times 1$ FC problem

The questions that we will pursue concerning the switch's egress, i.e. the OQS, are related to:

1. Flow Independence; How can the correlation between flows competing for link access be expressed? Does this correlation play an important role? Is flow independence necessary in all network environments? What is the impact of the two major switch architectures (CIOQ and CICQ) on the independence of flows?

2. Switch Dynamics; In a system of changing input loads, how does the traffic behave after switching, i.e. at the output? What is here the impact of the two major switch architectures on switch dynamics?

7.2 Switch Dynamics and Fairness

The purpose of this chapter is two-fold: to conclude the discussion on CIOQ/CICQ switch architectures that was started in Section 5.6, and to justify the utilization of RR as data-path scheduler.

The data-path scheduler is responsible for allocating link bandwidth, i.e. which packet to transmit, and for assuring promptness, i.e. when to transmit packets [54]. We will discuss in this section data-path schedulers in the light of communication and multiprocessor environments. Furthermore, we will point out metrics to better evaluate a SF's behavior.

We will first give an abstracted view of a SF and present related work. We will then state the problems and come up with solutions.

7.2.1 System Description

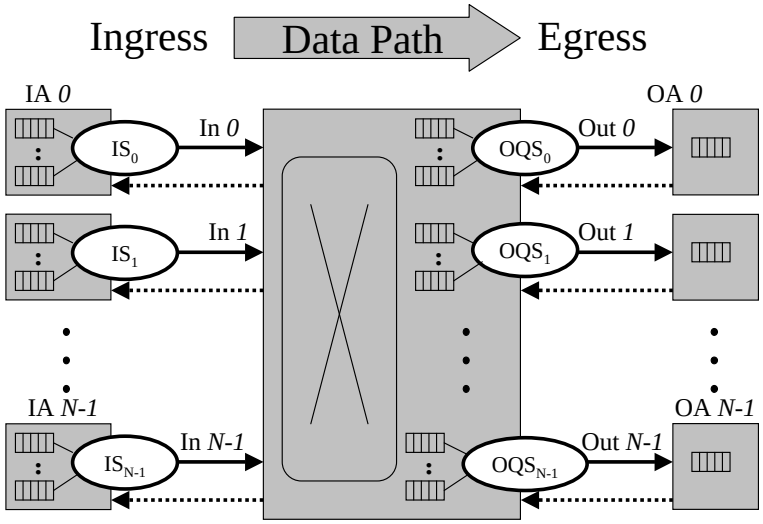


Figure 7.1: Important Scheduling Points in a Switch Fabric

Fig. 7.1 shows a generic SF with data-path schedulers at the ingress (IS) and the egress (OQS). The ingress schedulers serve the virtual-output queueing structure at each input. The ingress schedulers operate independently from each other, i.e. the scheduling decision of one IA is not known at any other IA. This is also referred to as distributed switch architecture [96] and in our terminology a d/c switch. In Fig. 7.1 we can see that there is one OQS per switch output port, in total N OQS per switch, all operating independent from each other. It is important to see that the OQS of one stage could be already the IS of the next stage, for instance in multi-stage fabrics. Therefore, the results of the OQS apply as well to the IS and we broadly refer to both as data-path schedulers, which have to have the same properties. Each OQS has access to one or multiple dedicated (output) queues and performs one read operation per packet cycle. For the sake of sequence preservation each such queue uses a FIFO service discipline. Various data-path scheduling strategies can be employed. We limit ourselves to a RR and an OCF scheduling algorithm for the reason that, without queue reordering, RR emulates a CICQ and OCF a CIOQ switching architecture. The RR-OQS has to make a selection from N output (cross-point) queues, whereas the OCF-OQS just reads from one global output queue. This is independent of how the memory is allocated to those buffers. It could be shared, or dedicated.

In order to support p QoS traffic classes, the number of queues in the switch is multiplied by p in both architectures. The CICQ switching architecture would therefore do queueing per

input per QoS class, whereas a CIOQ switch would perform queueing per QoS class per output. In both cases the support of bandwidth guarantees could be realized by a WRR scheduler, while for a guarantee of delays a weighted fair queueing (WFQ) [59] algorithm could be chosen. For no guarantee a simple RR is appropriate. Such an QoS-enhanced OQS would therefore make its selection from p queues in the CIOQ and from Np queues in the CICQ case.

Fig. 7.1 is similar to the Reference switch model presented in [63], whereby selective backpressure, a stateless FC scheme, is used to prevent output queues from overflowing. In our analyses a credit-based FC is applied. The authors of [63] apply queueing per QoS class at the output, an approach whose validity we want to check in this section.

Also, a similar model is used in [48], consisting of a CICQ switch which implements distributed fair queueing . Those authors have recognized the need for QoS support within a switch and propose an elegant solution. However, as their algorithm requires the exchange of the virtual time between IA, switch, and OA, the "FC-overhead" is large, something that we want to avoid. Furthermore, their analysis neglects the RTT between adapters and switch.

7.2.2 Background

Scheduling the data-path received extensive attention in the literature. However, it is mainly discussed in the context of routers and protocol processing, which we briefly described in Section 2.1. The discussion is often neglected at the level of the switching, see for instance [91].

We will therefore point to the issues in router scheduling that apply to scheduling the data-path of the switch. The discussion centers on 1) fairness and 2) implementing QoS aspects. We recapitulate by putting more weight on the former topic.

Fairness Issues

The problem was recognized that if multiple sources compete for link access one flow may arbitrarily increase its share of the bandwidth, thereby negatively affecting others. Already in [54] it is discussed that a first-come-first-serve (FCFS) queueing algorithm is not appropriate to isolate flows. An ill-behaved flow can capture an arbitrarily high fraction of the bandwidth of the outgoing link at the expense of other flows. Demers [54] concludes that more "discriminating queueing algorithms must be used ... in *non-cooperative* environments". Nagle [55] proposed to maintain separate queues for packets from individual sources, whereby the queues are serviced in a RR fashion.

This was found to be an appropriate solution as it provided a fair share of bandwidth to every flow, except that it ignores packet lengths. Nagle's scheme assumes that the average packet size over the duration of a flow is the same for all flows. Time has shown that this is an invalid assumption. A flow using long packets would receive more bandwidth than a flow using short packets.

Demers proposes an ideal algorithm, a bit-by-bit round-robin (BR), which solves the (Nagle's) problem of unfairness due to variations in packet sizes. As BR is impractical to implement Demers further proposes a fair queueing (FQ) algorithm that is able to approximate the ideal BR behavior. By sending the packet with the shortest finish time first, queue re-ordering is necessary. Thus FQ solves the bandwidth fairness problem independent of a variation in packet sizes, but it has a complexity problem. It is hard to implement at high speeds because of its time overhead of $O(\log n)$ for insertion of a packet into a sorted queue, whereby n signifies the number of flows.

McKenney [56] addresses the inefficiency of Nagle's algorithm with stochastic fair queueing (SFQ) that uses hashing to map packets to corresponding queues. An advantage is that it

requires considerably less resources (queues) than flows that are supported. It also reduces the overhead to $O(1)$. Its main deficiency is that it is unfair, when flows collide.

An algorithm called deficit round robin (DRR) was proposed by [57] that keeps the properties of FQ by reducing the complexity to $O(1)$. It emulates the original FQ algorithm by selecting packets based on deficit quanta of flows.

From this discussion we identify the following problems. There is an issue of fairly allocating bandwidth, which has a dependency on packet size. This problem arises in particular in non-cooperative environments [27]. Moreover, resource consumption in number of queues, and complexity in processing time required to perform the algorithm play an important role. These issues must carefully be considered when designing queueing algorithms.

As we assume fixed-sized packets as the entity of switching in our SF, we can neglect in our case the solutions considering variable sized packets. Therefore, to provide fairness as in [55], it is necessary and sufficient if sources queue separately and a RR is used.

We will now apply this result to a SF. We will start from the switch's output. Independent of the buffer organization of the switch, each OQS has to serve traffic originating from N inputs. It is important to see that each input adapter connects to an independent part of the network. This independence may be justified by different protocols, and hence various end-to-end or network-access level FC mechanisms that are in place. From this perspective the environment is non-cooperative. Even if all sources applied the same standards and protocols, isolation is still a requirement, as compliance to standards is not assured and sources might maliciously alter algorithms to improve their performance. These are the reasons given by [54] to justify the isolation of flows. We want to add two new aspects.

The first aspect applies to locality of flows. In a communication systems environment, IAs serve flows from different locations. These flows are to be seen independent from each other. They all compete for output link access and have therefore to be protected against flows intentionally or unintentionally exceeding their share. Hence, we not only see the fairness issue due to a difference in mechanisms, but also due to locally disparate sources. We will analyze this more closely in Subsection 7.2.5.

The second aspect relates to the notion of "non-cooperative". The above is true, if we consider the sources greedy, or not trust-worthy, i.e. non-cooperative. But what, if the switch operates in an environment, where this underlying assumption is not true? A *cooperative* environment is for instance given in a multi-processor system. Here, multiple processing nodes work hand in hand on the computation of a result. This is to be achieved as fast as possible. Hence, in such an environment, the FC mechanisms are identical, and no source will unruly exceed its bandwidth share, because it could hinder the fast computation of the result. From this perspective, we advocate for OCF-OQS in such environments.

QoS Aspects

QoS aspects are centered around guaranteeing bandwidth and transmission delay. The WRR is being used to deliver bandwidth guarantees. Also, algorithms have been proposed based on FQ that are able to guarantee delays, such as weighted fair queueing (WFQ) [59], which combines FQ with a leaky bucket admission policy to provide end-to-end latency bounds. We have already pointed out how these QoS aspects and the related QoS classes can be realized in switches.

7.2.3 Problem Statement

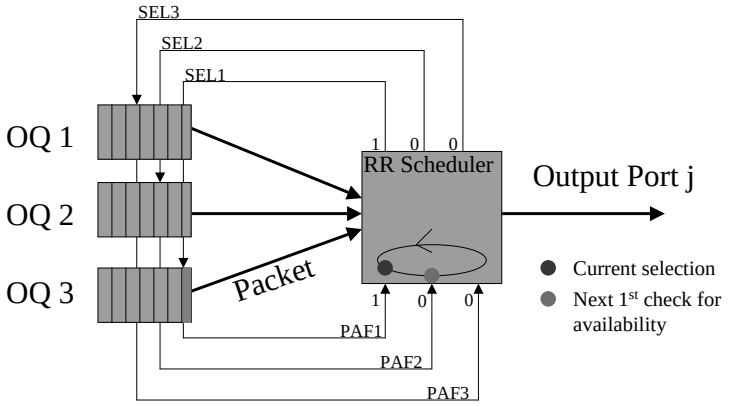
From the previous section the following issue needs attention: While in the past a CIOQ switching architecture was predominant, an architecture whose outputs are typically served in an OCF manner, research results of scheduling algorithms in the router arena mandate a RR-OQS[1], an algorithm that intuitively will be applied at the output in a CICQ switching architecture. With the results on router data-path scheduling reviewed in the previous subsection, the question arises how fair the service is that an OCF provides or, equivalently, how fair the output service of an CIOQ switch is. What metrics do we have to apply to render the differences visible?

We saw that for routers a non-cooperative environment is assumed. It is characteristic of communication systems. If this assumption is not valid, and instead a cooperative environment is assumed, then RR no longer is first choice. A cooperative environment is for instance represented by a multi-processor system.

In the remainder of the section, we will therefore:

- show that using not appropriate switch metrics, the differences between CIOQ and CICQ are hardly visible
- propose appropriate metrics
- determine the degree of input fairness (static behavior) of RR and OCF
- analyze the dynamic behavior of the RR and OCF algorithm
- assign an OQS to the environment were it seems most promising
- give reasons why we prefer a RR service discipline for all data-path schedulers in our switch model

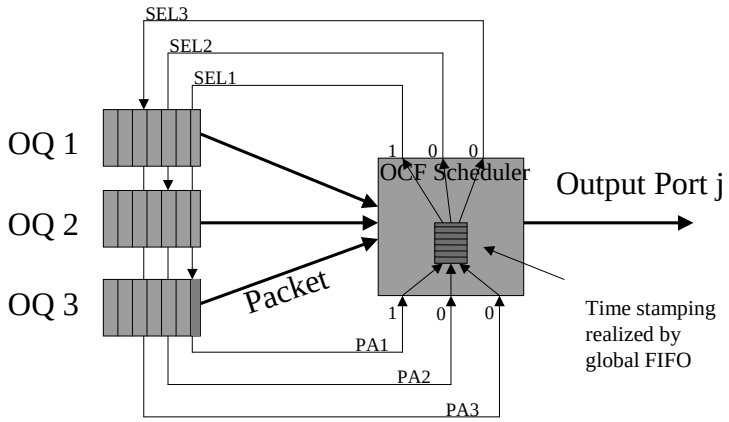
7.2.4 Experimental Set-up



SEL … Select Signal
PAF … Packet Available Flag

Figure 7.2: Simplified Schematic of a Data Path Scheduler Applying Round-Robin

[1] assuming fi x-sized packets



SEL … Select Signal
PA … Packet Arrival

Figure 7.3: Simplified Schematic of a Data Path Scheduler Applying Oldest-Cell-First

The two schedulers that we study are displayed in Fig. 7.2 and 7.3. As we already pointed out, RR maps directly without any additional effort onto a CICQ, whereas an OCF onto a CIOQ switch architecture. In the RR scheme, the status of the queues is updated each packet cycle to the scheduler by a packet available flag (PAF). The scheduler has a pointer, which points to the last queue i that has been served out of a total of N queues. The next queue that will be visited first is queue $i+1$ modulo N. It is skipped if it has no packets eligible and the pointer visits the next queue until it has found a candidate to send or it has visited N queues. A queue will send a packet upon notification by a select signal, a '1' in Fig. 7.2.

In the OCF scheme, there is a packet arrival (PA) signal instead. It will be stored as a vector in a global FIFO. If n PAs are active simultaneously, the conflicts are resolved by storing n vectors with just one bit set. Here, the order could be for instance determined by increasing queue destination address. The global FIFO therefore realizes a time stamping of the PAF vectors. However, the time stamping is potentially unfair for simultaneous arrivals. One vector will be read per packet cycle and its contents is used as the selection signal. This OCF arrangement is suitable to compare the two scheduling algorithms, because it allows us to analyze the N queues requesting service.

In order to study the different behavior, we exemplarily take three independent flows labelled "A", "B" and "C"with destination "D". There is no FC mechanism activated and the queues are of infinite length. The experimental set-up is such that the same flow of randomly generated traffic is duplicated and directed to a set of queues that is served by OCF, and another set of queues that is served by a RR scheduler.

We will measure the aggregate mean delay, as well as the mean delay per flow and the corresponding momenta over a period of 210,000 packet cycles. As a metric to express potential fairness or unfairness we propose to use the power metric that is defined as the ratio of throughput to delay according to for instance [97]. A higher power is better, as either delay is lower or throughput is higher, or both. We will calculate the *aggregate power*, which is the sum of power of all three flows A, B, and C to compare the quality of the service disciplines with respect to fairness. This service discipline that achieves a higher aggregate power value is said to be superior. This metric is expressive and easy to calculate. Moreover, the notion of sessions – a finite flow – used for instance in the max-min fairness calculation, see for instance [58], is not required.

We analyze three different scenarios. In one scenario the sum of all three contributing flows is 100%. In the other cases the sum is more than 100%, i.e the output link is over-subscribed.

7.2.5 Results

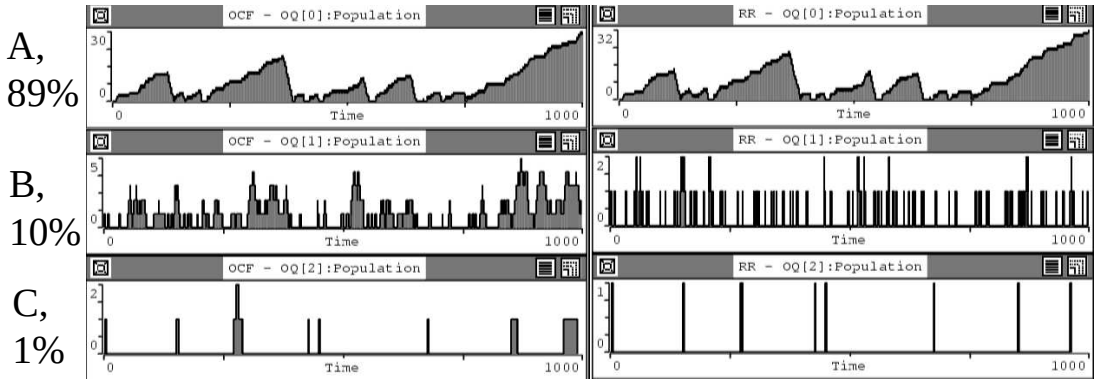


Figure 7.4: A, B, C Send 89%, 10%, and 1% of (Output) Link Capacity to D

Table 7.1: Measured Delay Statistics of 89/10/1

	OCF			**RR**		
A	Mean: 125.3 Min: 1	StDev: 88.1 Max: 386	Power: 1491.6 COV: 0.70	Mean: 140.5 Min: 1	StDev: 98.6 Max: 420	Power: 1330.3 COV: 0.70
B	Mean: 125.1 Min: 1	StDev: 87.6 Max: 387	Power: 167.9 COV: 0.70	Mean: 1.142 Min: 1	StDev: 0.41 Max: 6	Power: 18404.9 COV: 0.36
C	Mean: 123.7 Min: 1	StDev: 87.8 Max: 378	Power: 167.9 COV: 0.71	Mean: 1.111 Min: 1	StDev: 0.33 Max: 4	Power: 1890.1 COV: 0.30
Total	Mean: 125.3	StDev: 88.1	Power: 1827.4	Mean: 125.3	StDev: 102.7	Power: 21625.3

The traffic scenario in Fig. 7.4 is such that input 1 delivers 89% load of bursty nature (A), while input 2 additionally offers 10% (B), and input 3 another 1% (C). Therefore, the output link is not over-subscribed. Due to traffic randomness there will be contention. The questions is how the three different flows are treated and how fair this is being considered. The figure displays the queue length over time from start of simulation to an arbitrary end-time – here $t = 1000$.

The different behavior of OCF versus RR becomes apparent, when looking at Figure 7.4 and the statistics in associated Table 7.1.

OCF yields an equal delay distribution of about 125 packet cycles, with standard deviation of approximately 88. It is to be noted that the numbers for the three individual flows is in the range of the aggregate delay values. The aggregate power value calculates as 1827.4. with the detailed power values for the single flows noted in Table 7.1.

On the contrary, RR treats the three flows differently. While the mean delay of flow from A increases to about 140 with a higher standard deviation of 99, the lightly loaded flows from input B and C have much better characteristics. The mean delay for those flows is about 1 with a standard deviation of 0.4 and 0.3, respectively. The significant differences in delay were not to be observed, if just the aggregate values were captured. Judging just from the performance

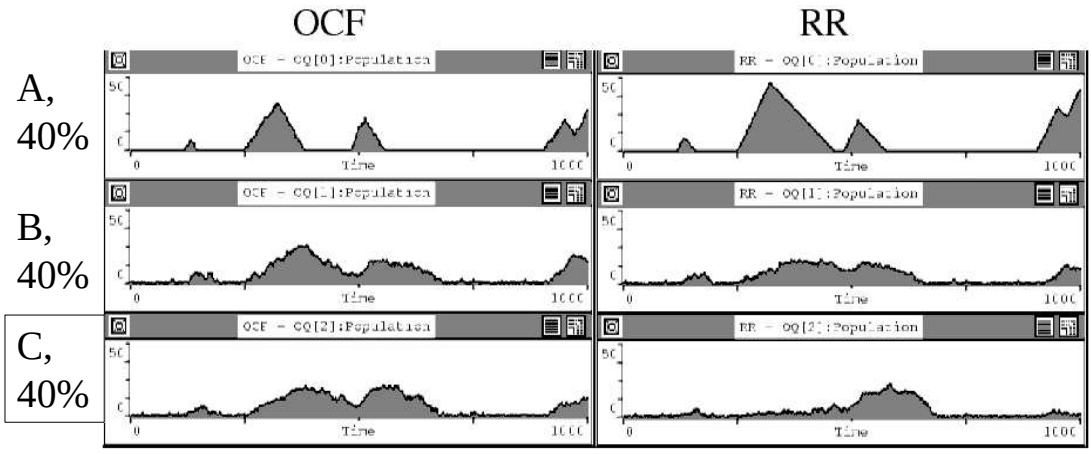


Figure 7.5: A, B, and C Send 40% Bursty, 40% Non-bursty, and 40% Non-bursty Traffic to D

aggregate, OCF would be superior. However, from a per-flow perspective RR is more fair. This is expressed by the aggregate power metric that is with 21625.3 significantly higher than in the OCF case. The independence of flows is statistically expressed in the coefficient of variation (COV), which is only about 0.3 for flows 2 and 3 in the RR case, whereas they are around 0.7 in OCF.

The queueing behavior is also distinguishable. In OCF, there is queueing observed for both the 1% and the 10% flow, i.e. C and B. This is not the case in RR . Both flows always receive their share of link bandwidth and therefore do not queue up. This approximates the max-min fairness properties as for instance defined and evaluated in [58].

The traffic scenario in Fig. 7.5 is such that A, a bursty source, sends to D 40% of the output link capacity. The non-bursty sources B, and C also send 40% of the output link capacity to D. The output link is over-subscribed, since the average capacity is 120%. We can see in Fig. 7.5 how the RR stops A, the bursty source, from “stealing” away bandwidth from the others: the slope of the queue occupancy is steeper for RR than for OCF. At the same time, the queueing of the competing sources B and C is less severe. In Table 7.2, we have the entire set of statistics listed. What we already observed in Fig. 7.5 graphically, is here expressed by statistical values. For RR the mean delay is larger for A, whereas for B and C, it is smaller. Also, the average power is slightly better for RR than for OCF.

Table 7.2: Measured Delay Statistics of 40/40/40

	OCF			**RR**		
A	Mean: 11001	StDev: 6264	Power: 6.3	Mean: 14550	StDev: 7952	Power: 4.8
	Min: 1	Max: 21435	COV: 0.57	Min: 1	Max: 26339	COV:0.55
B	Mean: 11128	StDev: 6408	Power: 6.2	Mean: 9399	StDev: 5567	Power: 7.4
	Min: 1	Max: 21435	COV: 0.58	Min: 1	Max: 18956	COV: 0.36
C	Mean: 11185	StDev: 6361	Power: 6.2	Mean: 9256	StDev: 5630	Power: 7.5
	Min: 1	Max: 20694	COV: 0.57	Min: 1	Max: 19047	COV: 0.61
Total	Mean: 11103	StDev: 6361	Power: 18.7	Mean: 11049	StDev: 6921	Power: 19.7

The third traffic scenario in Fig. 7.6 is characterized by a bursty A source that sends at 50%. B transmits 50% non-bursty, which is also true for C sending 5%. The total average required

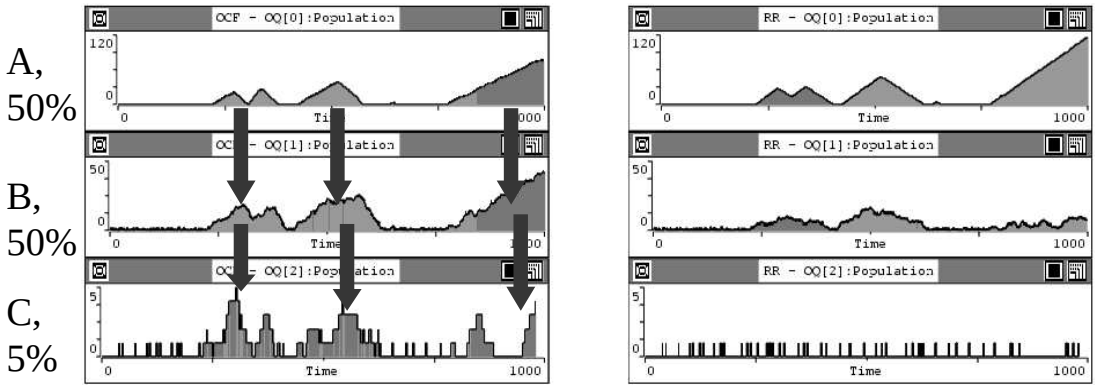


Figure 7.6: A, B, and C Send 50% Bursty, 50% Non-bursty, and 5% Non-bursty Traffic to D

link capacity is therefore 105%.

This scenario exemplifies the typical OCF behavior, which tends to attract traffic in neighboring queues. Indicated by the arrows in Fig. 7.6 we see that a burst of one source causes traffic to be stored in queues of other flows thereby affecting their properties. As a result, the OCF achieves better aggregate performance values (mean delay and delay variance). The delay variance and often the mean delay are smaller than compared to RR . However, the correlation between the flows is quite high. For C the COV is found to be around 0.64 when using OCF, whereas it is only 0.29 when using RR. It means that the flow using less bandwidth than it could (it could use up to 33% in our example) is better protected against "malicious" flows that exceed their share, such as B and A in this case. The average power is again significantly higher for RR than for OCF, which expresses a better and fairer service for all individual flows.

In Fig. 7.7 we see the delay distribution for the given scenario. As there is no FC employed, we see an unbounded queueing delay for A and B, which both exceed their 33% share. The important difference between the RR and the OCF scheduling is seen for the C flow of data. Using RR scheduling shows a clear delay bound, which is not the case for OCF. For the purpose of achieving delay bounds in OCF, the pointers in the global FIFO were required to be reordered. As we have discussed in Subsection 7.2.2 this is a time consuming process, which is unwanted. For the purpose of guaranteeing delay bounds, a FC mechanism had to assure that a maximum of queued packets was not exceeded. Such a functionality could be for instance incorporated into the RXS. However, a detailed analysis is beyond the scope of this thesis.

Table 7.3: Measured Delay Statistics of 50/50/5

	OCF			**RR**		
A	Mean: 5870.3	StDev: 3711	Power: 1699.2	Mean: 7397.5	StDev: 4997	Power: 1348.4
	Min: 1	Max: 12314	COV: 0.63	Min: 1	Max: 16678	COV: 0.67
B	Mean: 5796.5	StDev: 3736	Power: 1720.9	Mean: 4801.8	StDev: 2888	Power: 2077.3
	Min: 1	Max: 12314	COV: 0.64	Min: 1	Max: 9715	COV: 0.60
C	Mean: 5823.9	StDev: 3793	Power: 1.8	Mean: 2.67	StDev: 0.76	Power: 3932.6
	Min: 1	Max: 12314	COV: 0.64	Min: 1	Max: 10	COV: 0.29
Total	Mean: 5833.4	StDev: 6361	Power: 3421.9	Mean: 11049	StDev: 4377	Power: 7358.3

We are now interested in the promptness of the two queueing service disciplines. We expose the two arrangements to changing load and are hence checking the dynamics of each queueing algorithm. We apply the following scenario: C always has 71.4 load. After 66 cycles, we enable A with 71.4% and B with 40%. In the 330th cycle, A is switched off. In this model we have

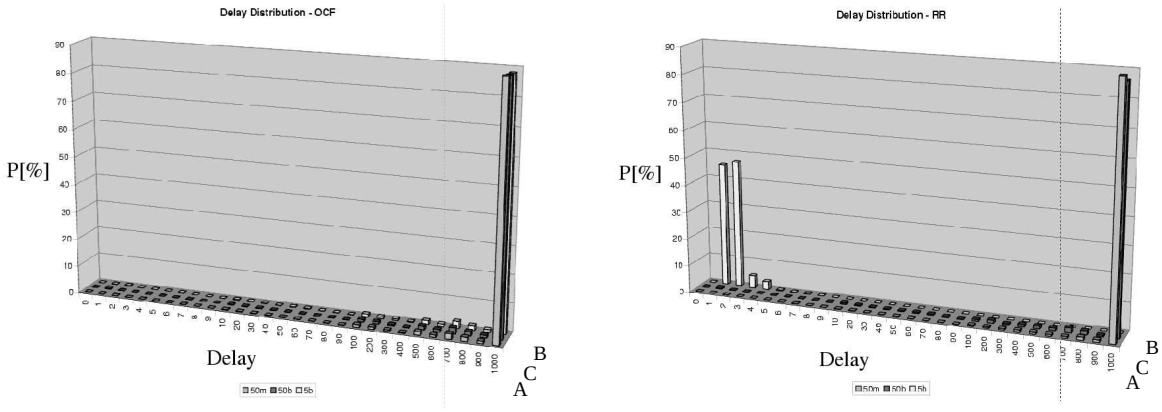


Figure 7.7: Comparison of Delay Distribution; (a) OQS=OCF, (b) OQS=RR

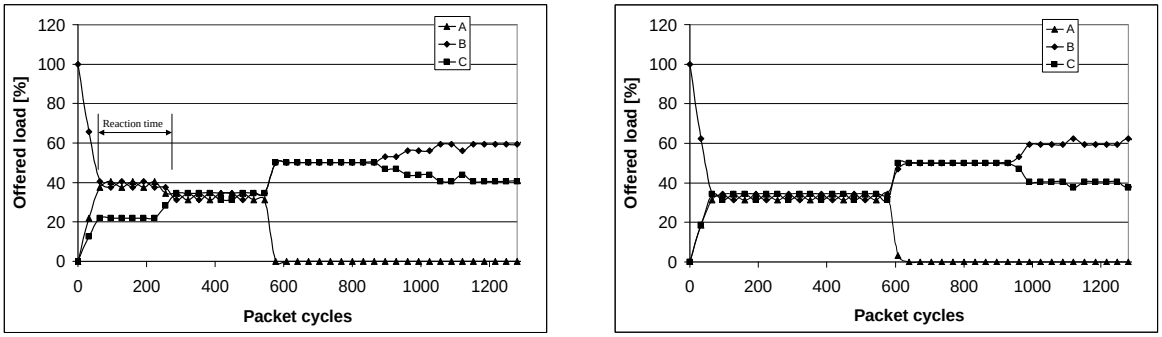


Figure 7.8: Comparison of Dynamics of a Switch; (a) OQS=OCF, (b) OQS=RR

enabled FC and use a RTT of 64, which corresponds to a link length of 32 (our design point). The throughput measurements of Fig. 7.8 are taken at the output link. We see C to occupy the output link immediately as it is the first flow to be active. Then A and B are enabled. We now follow the behavior for OCF in Fig. 7.8a) and for RR in Fig. 7.8b). As all sources send more than 33% of load during packet cycles 66 and 330, each should achieve exactly 33% of the available link capacity. While RR achieves this almost instantaneously, in OCF this goal is achieved 250 packet cycles later. After A is shut off and has drained all its remaining packets, in OCF the target of 40% for B and 60% for C are slowly approached, whereas in RR it is an instantaneous process. We see that RR has superior dynamics than OCF. It is better able to react to changing loads than OCF is. This demonstrates the different behaviors of the two scheduling schemes related to promptness. Equivalently, this is the difference in the behavior of CIOQ and CICQ switches under the assumption that queue re-ordering is not performed. We therefore conclude that the dynamics in CICQ switches are much better than in CIOQ ones.

7.2.6 Summary and Conclusion

A communication systems environment is non-cooperative. This calls for protection against competing flows that exceed their bandwidth share. This protection is to be implemented in switches, as an output port and its associated OQS serves N inputs, originating from distinct locations. It was already discussed in literature that for such cases for instance a RR is an attractive mechanism to arbitrate amongst the competing sources. This discussion is hardly found for switch architectures. A CICQ switch architecture provides a queueing structure that achieves the protection requirement without queue reordering. A CICQ is therefore well suited in such environments.

In a multiprocessor system we are dealing with cooperative environment instead. Therefore,

here it is advisable to use CIOQ switches. There is no competition amongst flows for output bandwidth, as all nodes are working on the solution of one problem. We relegate the proof of a CIOQ architecture to be more beneficial in such an environment to future work, because a simulation environment was needed that executed the program – or generated network messages – with the switching subsystem incorporated. A benefit can only be seen if the execution time of the program was reduced.

We analyzed the behavior of the OCF and RR queuing algorithms and made the following observations. The *aggregate* delay values, such as mean delay and delay variation are often misleading to evaluate performance. They fail to reveal the quality of service of individual flows. We therefore proposed the power metric for this purpose. It is simple to calculate and does not require the notion of sessions. It impressively demonstrates the differences in overall service quality. A good criteria of evaluating the independence of flows was found to be the COV. We refer to this as static behavior or input fairness.

At last the OQS's dynamics were analyzed and it was found that OCF has long reaction times to changing loads. For a non-cooperative environment, this is an unacceptable behavior.

We choose a RR-service discipline along all our data-path schedulers, because it is capable in delivering fairness and good overall performance results.

7.3 Initial Experiments on Credit Scheduling

In the following we are interested in the potential (merits), and risk of credit scheduling (drawbacks). Furthermore, we investigate in the occurrence of parallel departures which gives us a good indication of the utilization of a FC channel with a bandwidth capable in transmitting N FC events.

For analysis of the potential and danger of FC scheduling, we utilize a credit FIFO and return *one* credit per tu per FCD as shown in Figure 7.9. The length of the FIFO is determined by the worst case that all cross-point buffers M_j of a FCD are full and the packets stored leave the switch in parallel. Hence, there are N write accesses (credit in) and 1 read accesses (credit out) per tu for a period of M cycles. For our design point of $M = \mathrm{RTT}$ no new packets can possibly arrive during these M cycles. Therefore, the storage capacity, i.e. the length of the credit FIFO FL is found to be $FL = M(N-1)$ as indicated in Figure 7.9. For the time being, we do not consider the credit FIFO as a strategy to return credits, but as a means to store and to serialize the return of credits.

For the purpose of evaluating the potential, we introduce the term *credit contention*. A credit contention situation is said to exist if the average number of credits queued per FCD is larger than 1. In such a case credits are being backlogged. It could be that a first credit out of a sequence of credits stored in the credit FIFO makes a VOQ feasible which at the time of its arrival has no packets to send. To make the case more severe, a credit further back in the credit FIFO was needed by another VOQ to make an eligible packet feasible. If the order of credit return could have been reversed in this case, the waiting packet could have been served immediately, hence boosting performance. For this reason, we see an average credit FIFO population of a value larger than 1 as a potential for credit, i.e. FC scheduling and thus to improve switch performance. The average credit FIFO population gives us a good indication of the occurrence of credit contention. We analyze this in Subsection 7.3.1

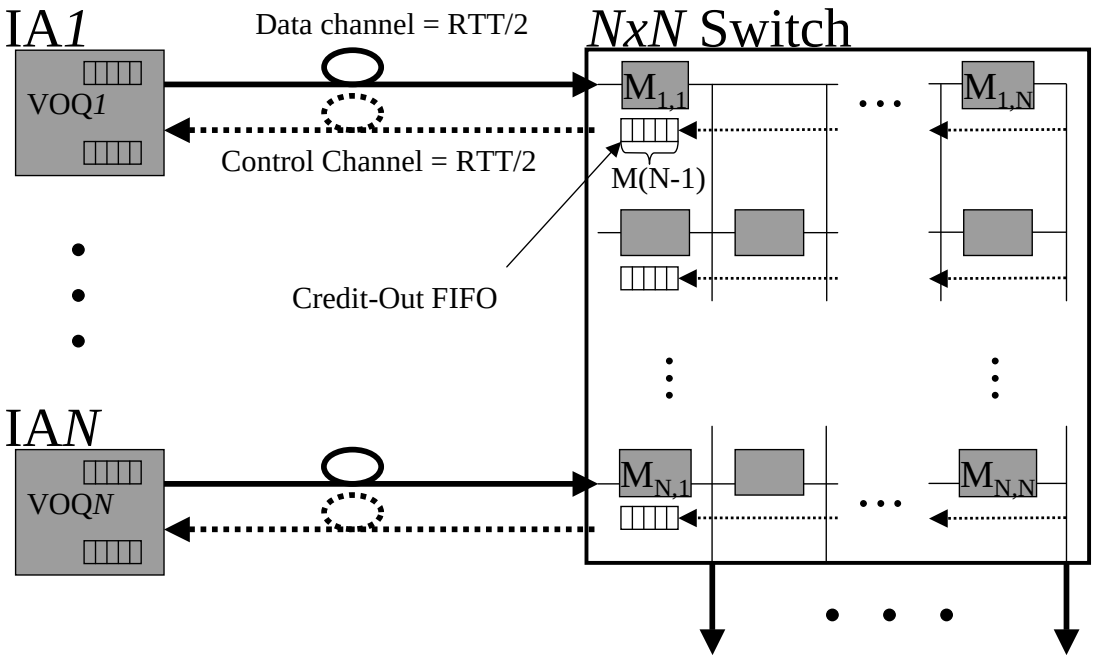


Figure 7.9: Experimental Setup for Initial Analysis on FC Scheduling

Credit contention is provoked by parallel departures. Therefore, for a switch of dimension N, there are N possible concurrent departures per FCD, resulting in N FC events. Provided a FC channel bandwidth that allows to return N FC events per tu, would this capacity be used? Consequently, what is the distribution of occurrences of departures out of one FCD? In Subsection 7.3.2 we evaluate the expected occurrence of parallel departures per FCD via simulations and analytical modelling using traffic with uniform destinations.

At last we may not forget overall performance. By reducing the amount of FC bandwidth to one FC event and possibly observing credit contention, credits for some FCSD may simply not circulate for a short period of time, thus possibly making eligible packets unfeasible. This may result in reduced performance. We therefore do a classical performance analysis investigating the delay and throughput in Subsection 7.3.3 comparing a $N \times N$ switch with a FC bandwidth of 1 and N FC events.

7.3.1 Potential of Credit Scheduling

Analyzing the average population of the credit FIFOs in a 8×8, 16×16 and a 32×32 switch subjected to bursty traffic with average burst sizes of 10, and 30 yields the results shown in Figs. 7.10 - 7.12. All system parameters used in this evaluation are summarized in Table 7.4.

All figures show the number of credits FL for input loads of $L = 80$, 95, and 98% and vary the cross-point memory sizes, from $M = 20\%$ RTT ($M = 13$) to $M = 200\%$ RTT ($M = 128$). We see that for high loads FL is not negligible. For instance, for $N = 16$, a memory size of $M = 64$ ($M = 100\%$ RTT), $BS = 30$, and an offered load of 95%, the length is on average 56.3 credits. At an offered load of 98%, this length even increases to 110.6, which is shown in Fig 7.11. Lower loads than 85% are irrelevant because we are interested in resolving output contention, which requires the provocation of contention situations, i.e. the operation close to or at saturation.

Additionally, all these curves show a converging behavior. For instance, the lines for an offered load of 80% and burst sizes of 10, and 30. Here, we notice that the saturation level of the number of credits FL queued in the FIFO is reached starting from a memory size of $M = 38$ ($M = 60\%$ RTT). For an input load of 95%, the curves saturate at around $FL = 25$ for $BS = 10$ and $FL = 70$ for $BS = 30$ starting from $M = 64$ ($M = 100\%$ RTT). For traffic with an input load of 98% the saturation is at around $FL = 60$ for $BS = 10$ and $FL = 160$ for $BS = 30$. There is no dependency of system size on the results for an input load of 80%. For input loads of 95, and 98%, saturation is earlier reached for $BS = 10$, than for $BS = 30$, because of more frequent on/off transitions. Depending on the input load and burstiness there is a maximum number of credits queued for traffic with uniform destinations.

For burst sizes of $BS = 10$ the length of the credit FIFOs is at least $FL \geq 5$ across all cross-point memory sizes, system sizes and input loads measured. For $BS = 30$ the length is at least $FL \geq 10$. These values demonstrate that scheduling of credits is worth investigating, which we will pursue in later sections.

The converging behavior suggests that the impact of FC scheduling will saturate for memory sizes larger than $M = 100\%$ RTT. The potential of FC scheduling is in the slope of the credit FIFO length. We will have to confirm this speculation later by means of simulations.

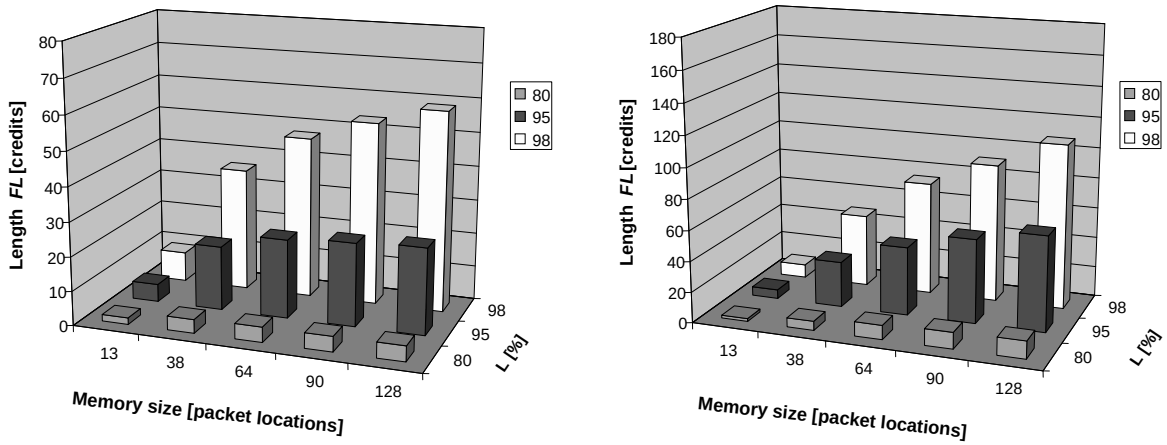


Figure 7.10: Average FIFO Population for $N = 8$; (a) BS=10, (b) BS=30

.

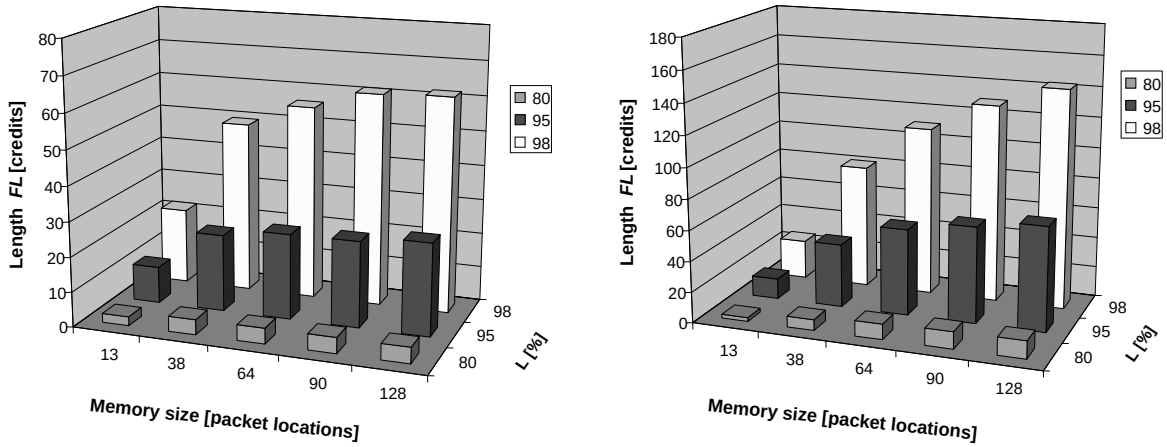


Figure 7.11: Average FIFO Population for $N = 16$; (a) BS=10, (b) BS=30

.

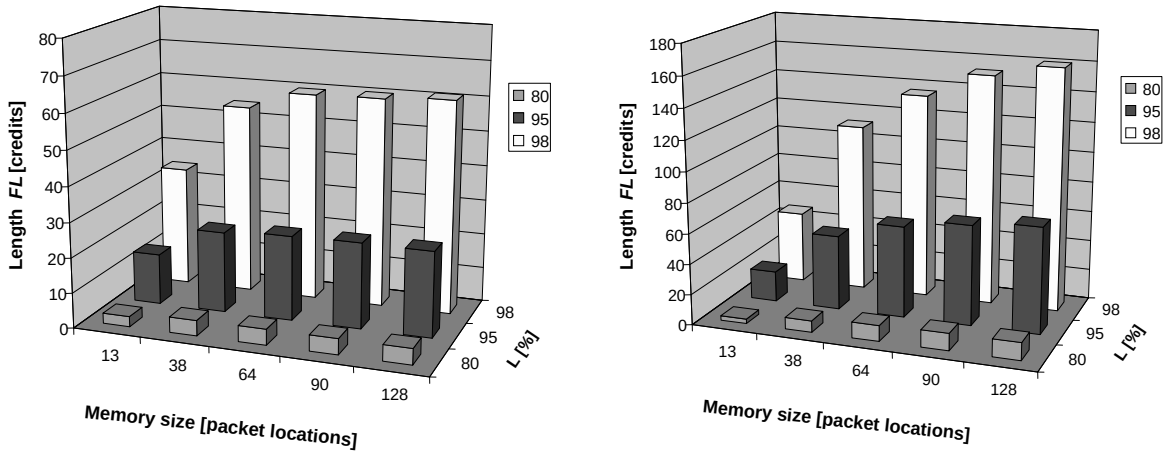


Figure 7.12: Average FIFO Population for $N = 32$; (a) BS=10, (b) BS=30

.

7.3.2 Parallel Departures

In this section we break down the number of departures per FCD per tu. Departures are of parallel nature if two or more packets leave a FCD simultaneously. We will use simulations as well as analytical modelling for traffic of uniform destinations to perform this evaluation. In previous chapters we have seen that new FC information to be conveyed is caused by packet departures. We would like to know the distribution of departures depending on input load,

port size and burstiness. This information is important in order to dimension FC bandwidth correctly and to identify possible system parameter dependencies. For instance, if there is a 90% probability for one departure per FCD, then providing FC bandwidth of N departures per FCD seems excessive.

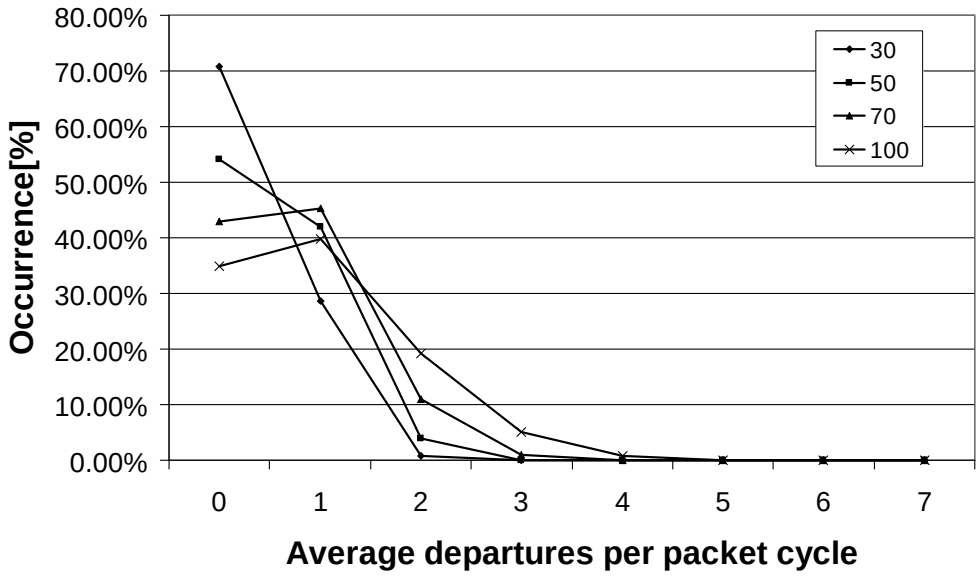


Figure 7.13: The Distribution of Output Departures

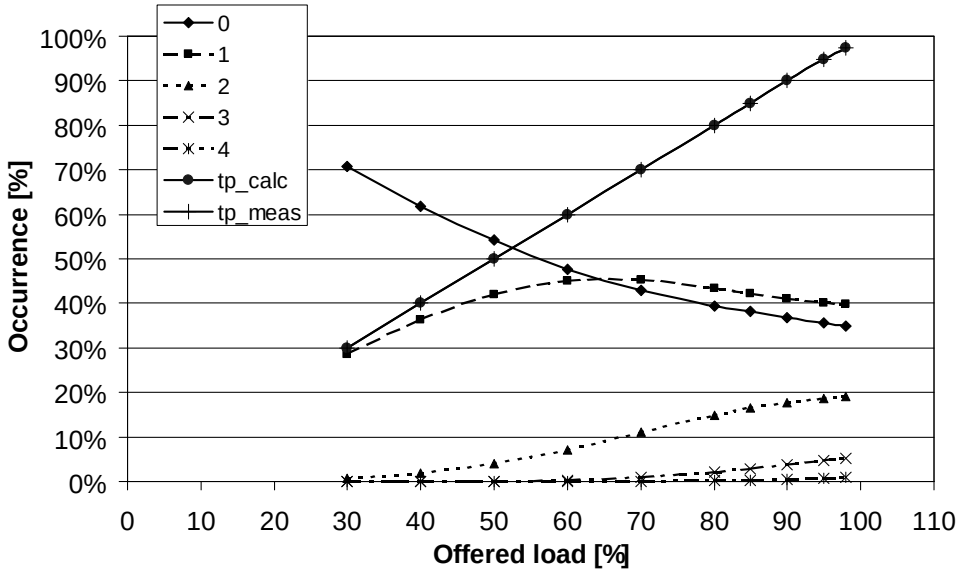


Figure 7.14: The Influence of Input Load on Output Departures

Simulations

Figure 7.13 explains the influence of the offered load to the occurrence of parallel departures. The results shown in this graph were collected using a $N = 8$ port device, $M = \mathrm{RTT} = 64$ and a traffic burstiness of $BS = 30$. The graph shows four different curves, for input loads of 30, 50, 70, and 100%. In order to be able to simulate a load of 100%, we stop simulations after 1,000,000 packets. We do three replications in this case.

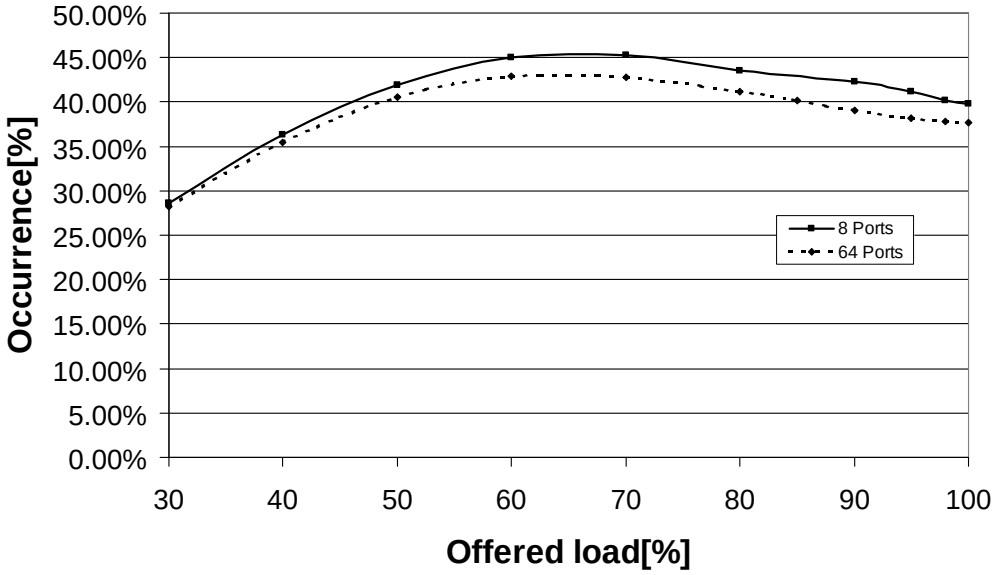


Figure 7.15: A Single Output Departure in Comparison for a $N = 8$, and 64 Port Device

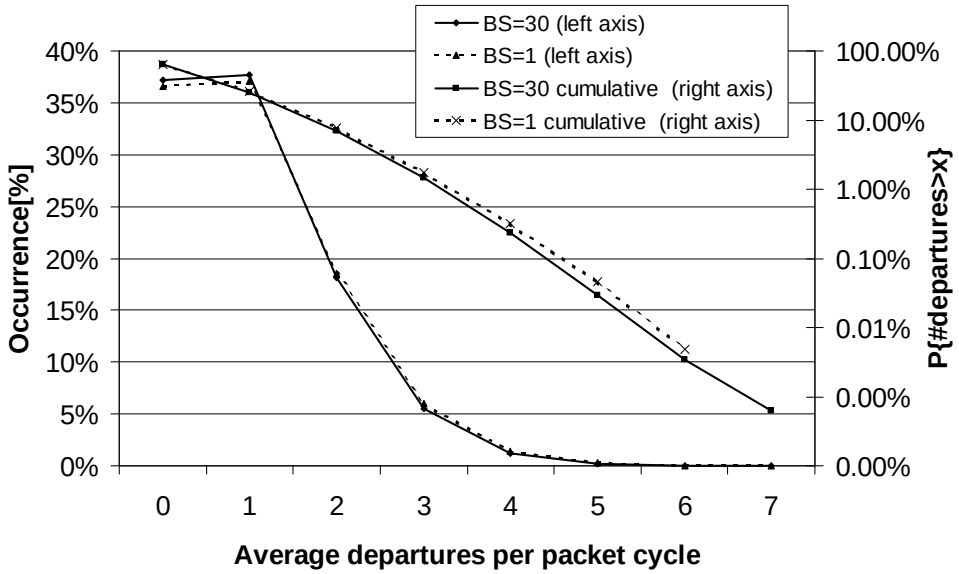


Figure 7.16: The Influence of Bursts on Output Departures

We see that for higher loads the probability of more parallel departures is larger. The probability of no departure from a FCD per tu drops for higher loads. However, it does not drop to 0%, but down to about 35%. If it would indeed drop to 0%, we had 100% departures of just one packet per tu per FCD. This would be the case when using directed traffic, i.e. one IA sends constantly packets to just one output (one-to-one flow). The parallel departures are therefore due to the randomness of packet arrivals. We further note that the probability of 5 of more parallel departures is close to zero. During our simulation periods we never noticed more than 10 parallel departures. This is of course simulation time dependent and not an upper bound. The upper bound is given by the number of FCSD, i.e. N.

A different perspective of Fig. 7.13 is given if the trend of 0, 1, 2, 3, and 4 departures from a FCD is plotted over the input load. This is done in Figure 7.14 where we see monotonic increasing parallel departures for higher input loads. The occurrence of 0 departures however

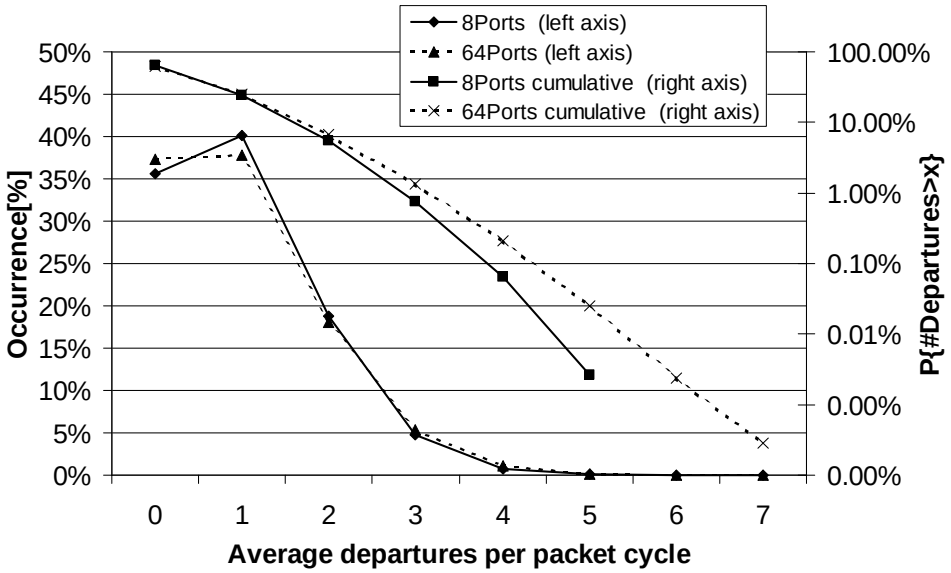


Figure 7.17: The Influence of the Switch Dimension on Output Departures

is monotonic decreasing from 70% down to 35%. The curve for one departure has a peak of $\sim$45% occurrence probability at an input load of $\sim$70% and ends at about 40% probability. In the same Figure 7.14 we have also plotted the measured throughput of the switching device (tp_meas). It was measured across all output ports. We demonstrate the probability of departures $p(i)$ weighted with the number of departures i yields the throughput $\rho = \sum_{i=0}^{N} p(i)i$. The resulting curve of is indicated as tp_calc and lays almost invisible on top of tp_meas in the Figure. Overall, the probability of parallel departures is proportional to the input load.

We compare the occurrence probability of one departure for a 8, and 64-port device in Fig. 7.15. We observe a 2-3% lower probability for loads larger than 50% for the 64 ports device. Less departures means less throughput. This results means that either the overall throughput of larger port devices for this type of bursty traffic is lower, or that this loss in performance is compensated by more parallel departures. This is to be analyzed next.

For larger switch sizes, there are more parallel departures to be expected on average than for smaller switch sizes. This is the result of Fig. 7.17 which uses $N = 8$ for small and $N = 64$ for larger switches. The probability of the occurrence of the number of departures is shown on the left y-axis. Here, the occurrence probability of 0 departures is higher, while the same probability for just 1 departure is lower for the $N = 64$ port device (dotted line). This observation suggests that there must be more information in the tail of this distribution, assuming equal or better throughput. This is clearly the case considering that with more ports, the switch also has more memory, which potentially increases performance. To better see that the $N = 64$ port device indeed has more parallel departures, the cumulative curves are shown. Its scale can be read from the right y-axis. The dotted line of the $N = 64$ device is higher than the solid line of the $N = 8$ port device, which reveals that the probability for more than x departures is also higher.

We conclude that the lower probabilities of one departure for a 64-port compared to a 8-port device are indeed compensated by more parallel departures.

The results of the influence of bursts are presented in Fig. 7.16 comparing burst sizes of $BS = 30$ and $BS = 1$ (Bernoulli traffic). The curve belonging the left y-axis shows the normal distribution of departures out of a FCD per tu, while the curve that belongs to the right y-axis

shows the cumulative result, i.e. the probability of more departures than indicated by the x-axis.

Bernoulli traffic shows more parallel departures than bursty traffic. This is surprising, because it implies that the best case traffic scenario for a switch's throughput, which is Bernoulli, has a worse behavior on the return of FC information, i.e. it requires more FC bandwidth. From a Bernoulli traffic perspective there is more FC information to be returned, because Bernoulli has the better switch throughput and thus more parallel departures on average. This is expressed by smaller amount of "no departures". Since at the same time there are less "one departure", the rest of the information must be distributed in more parallel departures, which can be verified in Fig. 7.16.

Analytical Modelling

We will show that the number of parallel departures per row can be approximated analytically for both uniform non-bursty and uniform bursty traffic, using Bernoulli experiments.

The binomial probability of at least i successes $P[X \geq i]$ is described by Bernoulli-experiments the following way:

$$P[X \geq i] = \sum_{k=i}^{n} \binom{n}{k} p^k (1-p)^{n-k}, \tag{7.1}$$

with p being the probability of success, k being the number of successes in n independent trials. Therefore, $n - k$ expresses the number of failures.

The analytical calculation is performed in three steps:

Step 1

In a first step, we are interested in the probability that one cross-point memory $M_{i,j}$ is occupied by arriving packets within $Q = N$ packet cycles at least once.

For a switch, the probability of success $p = \frac{1}{N}$, whereby N denotes the number of ports.

For non-bursty traffic, this probability calculates as follows:

$$P_{ber} = P[X \geq 1] = 1 - P[X = 0] = 1 - (1 - \frac{1}{N})^Q \tag{7.2}$$

For bursty traffic the number of effective packet cycles Q' that contribute to the equation is defined as

$$Q' = \frac{N}{B}, \tag{7.3}$$

whereby B denotes the average burst length. In this model, we assume $B \leq N$.

In such a case, the probability of success for at least one occupation reads as follows:

$$P_{bur} = P[X \geq 1] = 1 - P[X = 0] = 1 - (1 - \frac{1}{N})^{Q'} \tag{7.4}$$

For $Q' = 2$, which is for instance equivalent to $N = 64$ ports and a burst size $B = 32$, P_{bur} calculates as follows:

$$P_{bur}\Big|_{Q'=2} = \frac{1}{N}(2 - \frac{1}{N}) \tag{7.5}$$

Step 2

What is the probability that there is at least one entry in a column, based on the arrival probabilities p_{ber} of Eq. 7.2, and p_{bur} of Eq.7.4 and $Q = N$ calculated in step 1.

For non-bursty traffic, this can be written as

$$p_{ber,col} = P[X \geq 1] = \sum_{k=1}^{N} \binom{N}{k} p_{ber}^{k} (1 - p_{ber})^{N-k} = 1 - (1 - \frac{1}{N})^{N^2} \tag{7.6}$$

For bursty traffic, this calculates as follows taking Eq. 7.5 into account:

$$p_{bur,col} = P[X \geq 1] = 1 - (1 - p_{bur})^{N} = 1 - \left(1 - \frac{1}{N}(2 - \frac{1}{N})\right)^{N} \tag{7.7}$$

For $N = 64$, these probabilities yield $p_{ber,col} \approx 1$ and $p_{bur,col} = 0.86678$

Therefore, we have now an expression that tells us the probability that the OQS ("scheduler per switch column serving an output") has a packet to dispatch in this cycle. In this model we let the switch fill for Q cycles. Based on these "filling" probabilities, the OQS makes a selection with probability p_{rr} as detailed in step 3 and the probability for parallel departures is calculated.

Step 3

The probability of k parallel departures per row, requires N independent Bernoulli trials of the following nature for uniform non-bursty traffic:

$$P[X = k] = \binom{N}{k} (p_{rr} p_{ber,col})^{k} (1 - p_{rr} p_{ber,col})^{N-k}, \tag{7.8}$$

whereas p_{rr} is the probability of the RR output queue scheduler to select one M_i. $p_{rr} p_{ber,col}$ denotes the probability that the RR scheduler has selected this $M_{i,j}$, and a packet is also available.

For bursty traffic, this equation writes as follows:

$$P[X = k] = \binom{N}{k} (p_{rr} p_{bur,col})^{k} (1 - p_{rr} p_{bur,col})^{N-k} \tag{7.9}$$

Figures 7.18, and 7.19 compare the results from the simulations and analytical modelling for Bernoulli traffic ($BS = 1$) and bursty traffic with a burst size of $BS = 30$ using a small sized SF of $N = 8$. Figures 7.20, and 7.21 show this comparison for a large-sized SF, $N = 64$ and non-bursty and bursty traffic. We find a very good match for both curves.

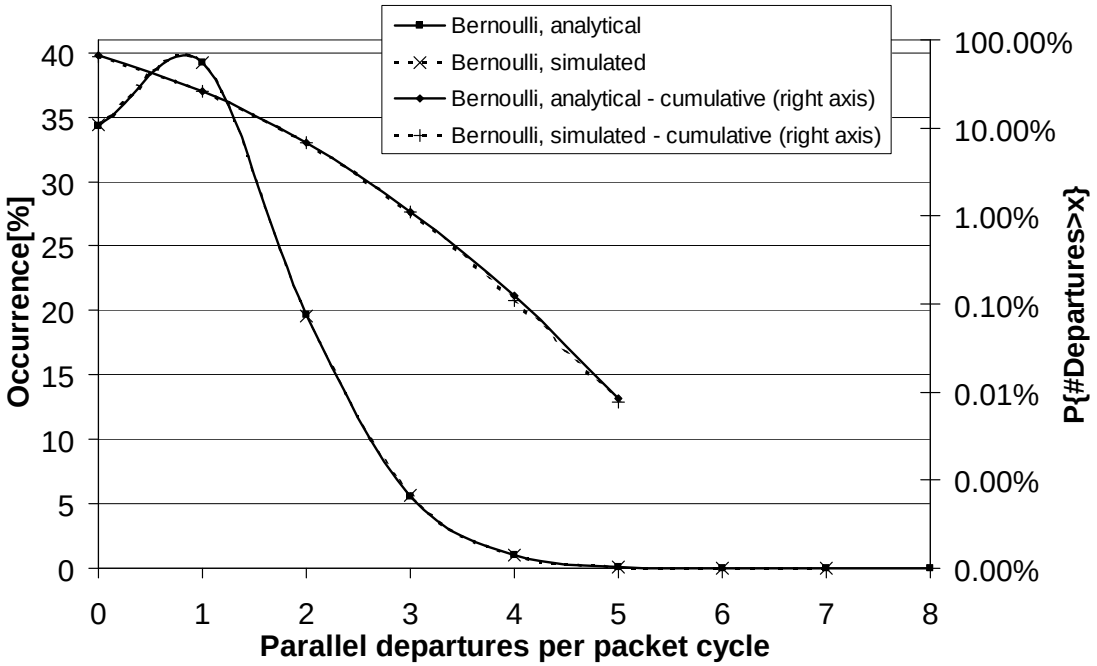


Figure 7.18: Comparison of Simulation and Analytical Modelling for $N = 8$, BS=1

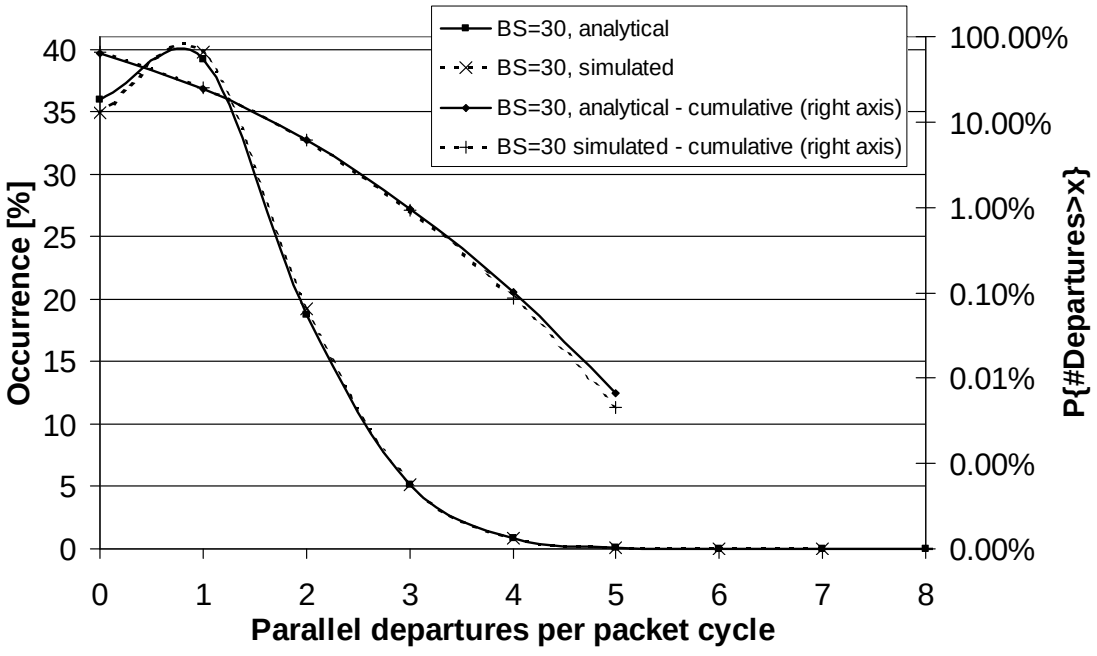


Figure 7.19: Comparison of Simulation and Analytical Modelling for $N = 8$, BS=30

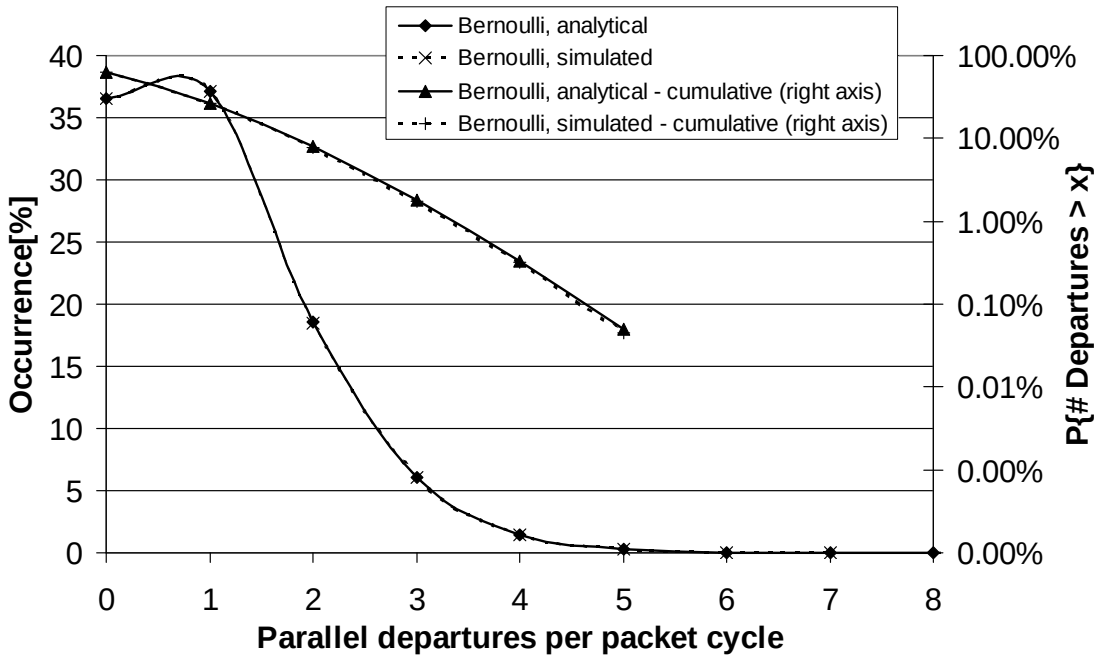


Figure 7.20: Comparison of Simulation and Analytical Modelling for $N = 64$, BS=1

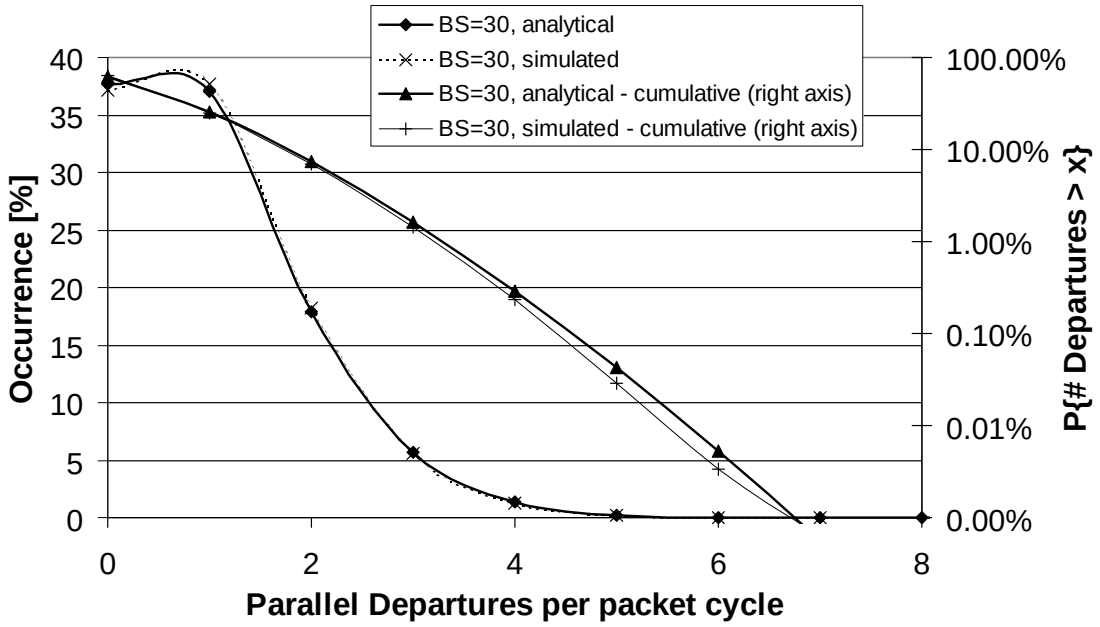


Figure 7.21: Comparison of Simulation and Analytical Modelling for $N = 64$, BS=30

7.3.3 Risk of Credit Scheduling

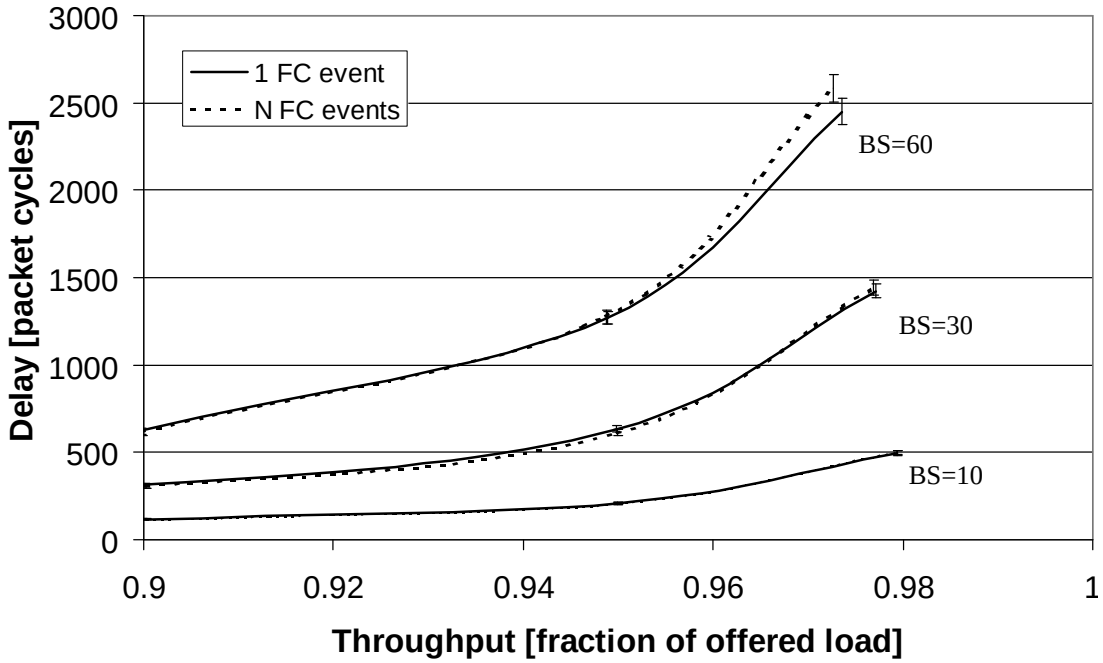


Figure 7.22: Comparison of a FC Channel with Partial Return of FC Information by Serialization and Complete Return of FC Information

The danger of credit scheduling is that returning just one FC event per packet cycle per FCD instead of N may harm performance, i.e. delay and throughput. The rationale of a loss in performance would be that credits temporary contending for reverse path access reduce the total amount of circulating credits per FCSD. As a result the performance of flows can not be maintained, hence reducing overall switch performance. In this subsection we disprove this argumentation by means of simulations.

We compare the switch in two configurations. In one configuration we store and serialize the occurrence of parallel departures by a simple FIFO and thus return only 1 FC event. In the other configuration we provide enough bandwidth for N FC events. The system parameters are a 16×16 switch fabric with an $\mathrm{RTT} = 64$ and a crosspoint memory size $M = 64$ (design point). As traffic parameters we choose bursty traffic with uniform distributions and burst sizes of $BS = 10, 30,$ and 60. Figure 7.22 plots the delay over throughput of these two FC channel configurations. The figure verifies that we may not expect any "danger", because both curves are in good accordance for all burst sizes investigated, as their 90% confidence interval is overlapping. The halfwidth of the error bar is 3%. Without statistical significance, we observe a slightly better performance at high loads and longer burst sizes for the case that the FC channel is able to return only partial FC information per FCD.

Credit contention is provoked by the random nature of traffic. It seems that owing for this randomness, a temporary reduction of circulating credits does not necessarily harm performance. Moreover, the generation of credits, which determines the order of returning credits is also random, as the selection of the OQS is not predictable.

7.3.4 Summary and Conclusion

Table 7.4: System Parameters of the Credit Contention Evaluation

RTT	M[%RTT]	System Size		
		8×8	16×16	32×32
RTT=4	20			
	40			
	60			
	100			
	140			
	200			
RTT=64	20	×	×	×
	40	×	×	×
	60	×	×	×
	100	×	×	×
	140	×	×	×
	200	×	×	×

In Table 7.4 we have summarized the system parameters under which the investigation on credit contention in Subsection 7.3.1 was performed. The FC bandwidth was reduced to just one FC event and serialization of the return of credits was achieved by a FIFO per FCD. As traffic parameters, we used bursty traffic with uniform destinations and burst sizes of $BS = 10, 30,$ and 60. We saw that credit contention is significant.

As a prerequisite of FC scheduling we need credits available at the beginning of the return path, from which a FC scheduler could choose from. Therefore, we introduced the term *credit contention* to give us an indication about the usefulness of our endeavor and to express the potential of FC scheduling. At this point in our investigation we can not say how many credits need to contend for reverse path access to make FC scheduling useful. However, a larger number of choices seems to be more attractive for a selection process. Therefore, the results show that FC scheduling is definitely worth further investigation.

Credit contention is evoked by parallel departures that occur randomly. The term parallel departures describes that two ore more packets leave a FCD concurrently within one tu. Hence, two or more credits are generated at once. In the evaluation of Subsection 7.3.2 we show the distribution of the average number of departures out of the switch's FCDs. This analysis gives an engineering perspective to the FC bandwidth problem. We see that a FC channel of a capacity of N events is not efficiently used independently of the burst or system sizes analyzed. During our simulations *we never observed more than 10 parallel departures* during a simulation time of 1,000,000 packets. Even if we simulated longer, the occurrence of higher parallel departures remained a rare event, which would not justify the unused FC bandwidth. This evaluation suggests that reducing the FC bandwidth is reasonable.

We also develop an analytical model to describe the phenomenon of parallel departures inherent to switching. The model achieves very good accordance with the simulation results for both varying switch size and varying burst size.

Along with the potential that we have looked at, there is the risk that comes from confining the FC channel to return one FC event per tu per FCD. The analysis performed in Subsection 7.3.3 demonstrates for bursty traffic with uniform destinations and a FIFO return that the FC bandwidth confinement does not result in worse overall performance. As metric we used delay and throughput. The system parameters are summarized in Table 7.5. Even though not

Table 7.5: System Parameters of the Evaluation of Partial/Full Return of FC Information

RTT	M[%RTT]	System Size		
		8×8	16×16	32×32
RTT=4	20 40 60 100 140 200			
RTT=64	20 40 60 100 140 200		$\times$	

statistically proven for high loads and a higher degree of burstiness, the results indicate that returning credits in a certain order does have an influence on switch performance. We will have to investigate this in more depth in later sections.

Reducing the FC bandwidth to 1 FC event is worthwhile because it saves in additional link cost. These initial experiments show that for FIFO return there is no loss in performance to be expected for traffic with geometrically distributed burst sizes and with uniformly distributed destinations (Subsection7.3.3). Furthermore, there is no necessity to provide a FC bandwidth of N FC events as the study on parallel departures in Subsection7.3.2 shows. As a confined FC channel provokes credit contention, a new way to return FC information is provided: FC scheduling. We have measured credit contention and concluded that there is potential for FC scheduling (see Subsection 7.3.1). FC scheduling seems promising and will be pursued in the next section.

7.4 The Reception Scheduler

Owing to the concept of autonomous FCDs, presented in Chap. 6, we can build the discussion about the RXS on a single FCD, but have in mind that it is applied to all ingress FCDs. We refer to the module capable of FC scheduling as RXS. There is one RXS per FCD implementing a return strategy as discussed in Section 7.5. We identify the single FCD shown in Fig. 7.23 as the $N \times N$ FC problem as described in Section 5.5. Whereas there we presented a more general description of the topic in the light of stateful and stateless FC schemes and their amount of FC information I and complexity C required, we will present here a more specialized view. We will lay out in the following how the RXS is embedded in this problem domain.

In Figure 7.23, we see the data-path interconnecting N send and N reception buffer pairs, hence interconnecting N FCSD. There is a feedback loop that originates from the RXS and uses the control path to return feasibility information to the upstream node. The time retardation of the feedback loop starting and terminating at the downstream node is RTT.

Each send and reception buffer pair maintains its own pool of credits. Its initial number is *invariant*. Therefore for any FCSD the number of packets on the data-path and in the reception buffer, together with the number of credits in the absorption pool, on the control path, and in the credit pool is constant.

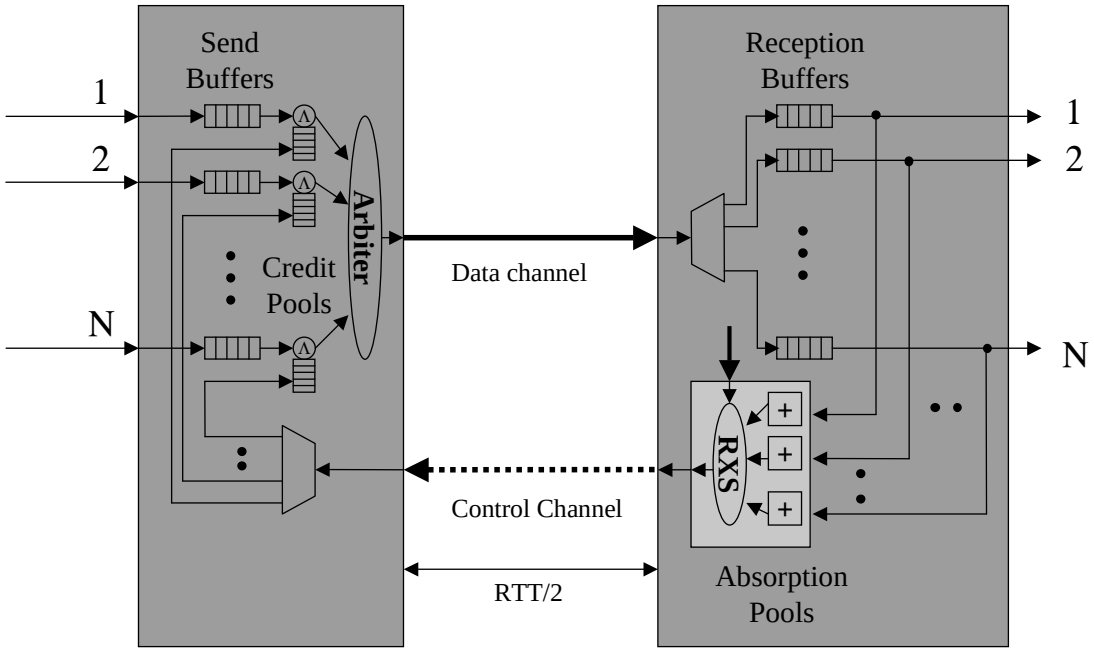


Figure 7.23: A $N \times N$ Credit FC with a Bandwidth-limited FC Channel

The feasibility information has to be distinguishable between sub-domains. This is achieved by assigning each FCSD a $\log N$ bits identifier, which is in the most common sense a *credit*. The amount of FC information I is a composition of n credits, hence $I = n \log N$. As we have seen in previous chapters, the maximum amount of FC information is $I = N$, whereby each bit position represents a reception buffer. A '1' signifies a credit, a '0' is no credit. For our analysis we will use the unconstrained FC channel, $I = N$, as our *Reference*, because all reception buffer state changes (FC events) are always returned without queueing time. As the amount of FC information translates into additional link cost, as shown in Subsection 5.1, which determines to a large degree the total switch cost, $I = N$ is impractical and we are therefore interested in the case of $n = 1$, hence $I = \log N$. Expressed in the number of FC events that may possibly be returned per tu, the Reference conveys up to N, while the FC scheme under study using the RXS returns only up to 1 FC event. As a consequence, some credits temporarily stored in the absorption pool will experience queueing time (credit contention). The reception scheduler uses this as a potential to prefer the return of some credits over others (see Subsection 7.3.1). Therefore this module tries to influence upstream data-path scheduling and hence future reception of packets, which explains its name.

We will now describe how the reception scheduler is embedded into the flow of information with Figure 7.23. The upstream scheduler selects one out of all eligible and feasible packets and dispatches it onto the data channel. Thereby, a credit is consumed. Each transmitted packet is therefore worth one credit. For the upstream, i.e. data-path scheduler, we assume a RR scheme to be in place. After RTT/2, the packet arrives at the current node and is stored in the reception buffer according to its destination address until it is scheduled for departure by the output queue scheduler (not shown in the Figure). For each destination there is an independent OQS in operation. Once a packet is scheduled and leaves the switch a credit is freed and stored in the corresponding *absorption pool*. The RXS has the possibility to consider the current reception buffer states of all FCSDs for its decision which credit to return next. This is indicated by the bold arrow to the RXS. The RXS takes only those absorption pools into account that have credits available. A credit is selected according to a RXS strategy, which we will deal with in

the next section. A selected credit arrives after RTT/2 packet cycles at the upstream node, where it is stored in the corresponding *credit pool*. Packets are feasible as long as there is at least one credit in the associated credit pool available. If a credit pool runs out of credits, we refer to it as *credit underflow*. Accordingly, if a reception buffer runs out of packets, we refer to it as *data underflow*.

7.5 Reception Scheduler Strategies to Prioritize Credits

The RXS-module may implement strategies of returning credits based on temporal, spatial or random properties. This gives us a general classification of RXS schemes. The number of credits to be returned is counted in a separate counter per FCSD, as indicated in Fig. 7.23 by the boxes with the "+" sign. This counter performs the bookkeeping of packet departures (add operation) and credit departures (subtract operation). It is referred to as the absorption pool, because it accumulates all credits that have been generated, but have not yet been sent.

The spatial strategies that we study are based on reception-buffer occupancy level, which in Fig. 7.25 is represented by the boxes containing a number. This number represents the current reception buffer packet occupancy level.

We investigate five strategies, one of random, three of spatial, and one of temporal nature. They will be presented after we briefly discuss our expectations.

Random strategy

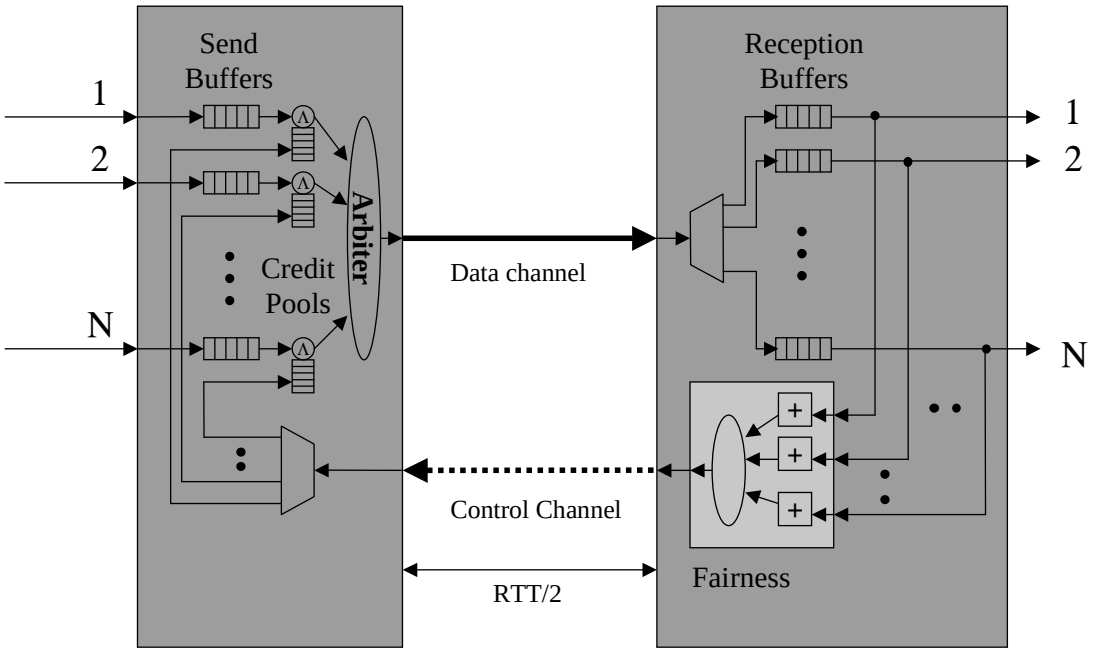


Figure 7.24: Random RXS Strategy

A strategy is considered random if there is no relationship between the generation of credits, the state of the reception buffers, and the issuance of credits. From this perspective, a RR return strategy based on the availability of credits as shown in Fig. 7.24 is random. The RR scheme is simply a means to provide fairness among credits competing for reverse path access. If our assumption that scheduling of credits results in a performance difference is wrong, then we should observe no difference to the other schemes.

Spatial strategies

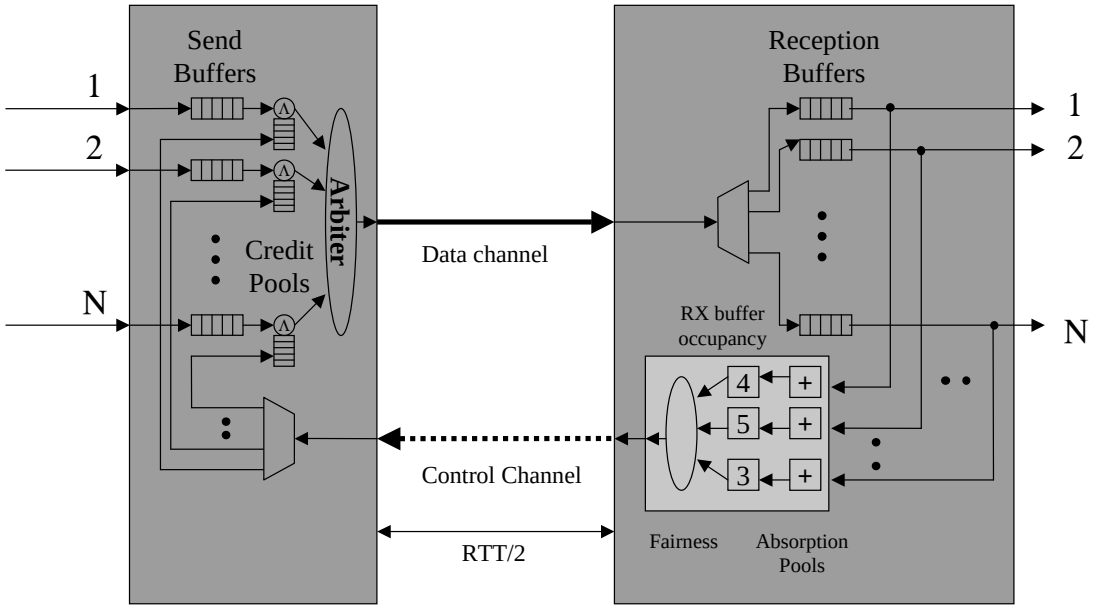


Figure 7.25: Spatial RXS Strategy

Spatial strategies are based on the reception buffer occupancy level. This is shown in Fig 7.25. There are several ideas that are believed to deliver performance benefits using this information. Each executes a certain strategy. The names are chosen to reflect the fact that the decision of returning credits is taken from the occupancy level of each memory.

The first spatial strategy, called *h*ighest memory *o*ccupancy *f*irst (HOF), focuses on keeping existing flows alive. It assumes a temporal continuity of flows at the sending side and tries to optimize upstream link utilization. HOF endeavors to deliver credits as fast as possible to those VOQs that will soon suffer *credit underflow*. The reasoning is that these reception buffers that have the highest occupancy also tie up at the reception side most of the credits materialized as packets. Therefore their corresponding send queues are likely to experience credit underflow, possibly leading to an idling data-path. The HOF strategy tries to avoid this situation and consequently favors the return of the credits of those FCSD whose reception buffer has the highest memory occupancy. A RR mechanism provides fairness in the case of equal memory occupancy.

The algorithm works as shown in Fig. 7.26 considering just six reception buffers that have credits to return. These buffers are shown in the top line. The small boxes here indicate the presence of packets, whereas the number indicates how many vacant packet locations the buffer still has. The first buffer from the left has 14 packets stored and 2 free locations. Each reception buffer has in this example a maximum capacity of $M = 16$ packet locations. The current buffer occupancy is available using occupancy counters per reception buffer. We start counting from zero and increment the counter values on packet reception and subtract on packet departure. These counter values are shown in the following line. The buffers with the highest occupancy are marked as winners, which is shown in the next line. For the case of multiple winners, a RR pointer is in place to make a selection. The final selection is shown in the last line. When numbering the reception buffers here from left to right, then a credit belonging to buffer number **four** would be returned in the next tu.

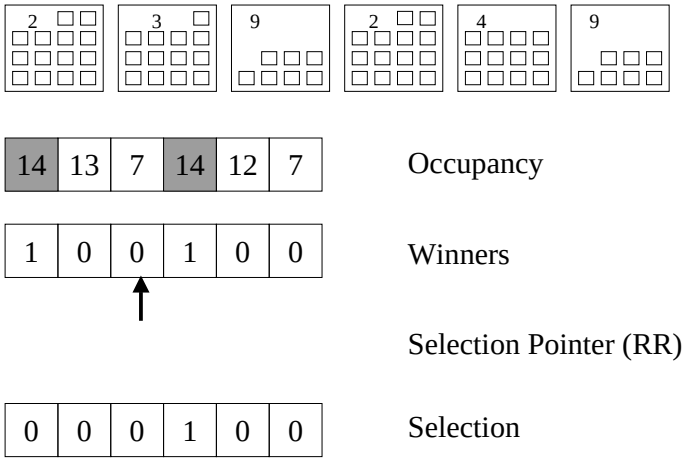


Figure 7.26: Highest Occupancy First (HOF) RXS Strategy

The second spatial strategy, called *l*owest memory *o*ccupancy *f*irst (LOF), focuses on supporting flows that make forward progress at the reception side. It assumes a temporal continuity of flows at the receiving side. LOF endeavors to return credits from those reception buffers first that are close to *data underflow*, as from the perspective of the reception side of the FCD, which strives for optimum performance, it is beneficial to fill its memory equally. The reception buffers that have a high occupancy level are likely to maintain a certain departure rate if the output scheduler allows it. Therefore, the return of these credits is not a priority. However, reception buffers that have a low occupancy must urgently return their credits because a certain output rate may not be maintainable. Moreover, reception buffers that have lower occupancy are more likely to make forward progress, because their output column is less contended. Therefore, the current forward progress of packets should be supported, meaning that these credits should be returned with priority.

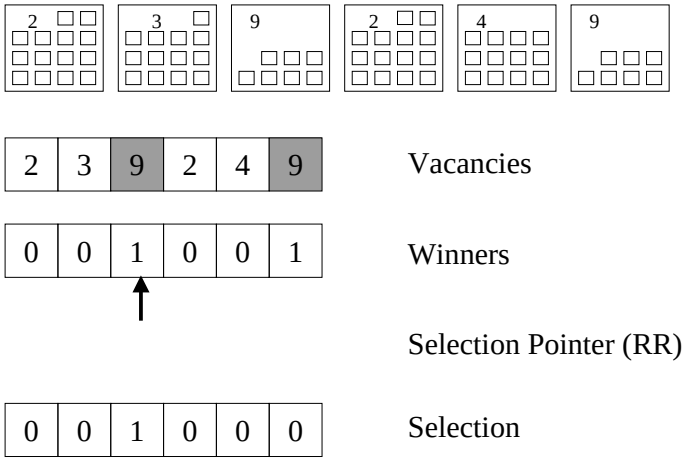


Figure 7.27: Lowest Occupancy First (LOF) RXS Strategy

The algorithm works as shown in Fig. 7.27 considering again six reception buffers that have credits to return. These are shown in the top line. Here we are interested in the vacancies, i.e. the number of free locations per reception buffer. We start counting from the maximum value of M free locations and subtract upon packet arrival and add one upon packet departure. The vacancies per FCSD are shown in the following line. The buffers with the highest numbers associated are marked as winners, as shown in the subsequent line. For the case of multiple

winners a RR pointer is in place to break ties. The final selection is shown in the last line. Numbering the reception buffers again from left to right, a credit belonging to buffer number **three** will be returned.

The third spatial strategy, called *l*owest memory *o*ccupancy *f*irst using the *c*ommunicated *l*ength of the *i*nput *q*ueues (LOF-CLIQ), additionally considers the state of the send buffers. The state of the send buffers is sent to the reception side and is preserved in a bit-vector until updated from the sender. This bit-vector is also referred to as the CLIQ-vector. The reasoning for this strategy is similar to the LOF strategy. However, here it is considered that the send queues may not have packets available at the time the receiver makes its decision according to the first scenario in Chapter 6. For instance, if the traffic pattern is such that one single output port is favored, then the plain LOF strategy would constantly return the "wrong" credits. This is because a favored port has to deal with most packet contention and the most contended outputs are not prioritized by LOF. With the knowledge of the input queue occupancy this can not occur. We consider this strategy to be more robust under changing traffic scenarios. CLIQ involves some overhead which we want to quantify next.

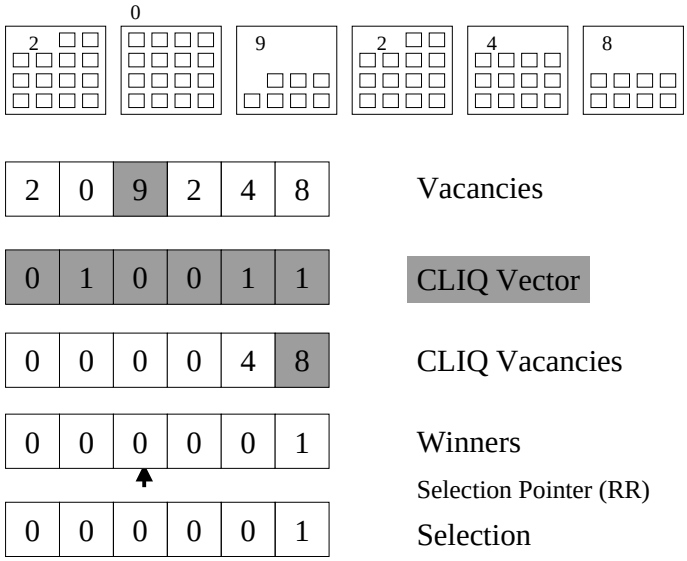


Figure 7.28: Lowest Occupancy First Considering the Communicated Length of the Input Queues (LOF-CLIQ) RXS Strategy

As the acronym CLIQ indicates, different lengths of the occupancy of the send queues could be communicated. For the moment, we only communicate whether an input queue is empty or non-empty. There is at most one arrival and one departure per IA per tu. This means that in the worst case one VOQ changes to non-empty ("first" packet arrives) and another one to empty ("last" packet departs). The latter case is covered by providing one bit in the header that indicates whether this packet was the last of its input queue (VOQ). This extra bit is sufficient, because the destination buffer is given by the packet address. Thus the corresponding entry of the CLIQ vector can be updated. For the case of communicating a state change to non-empty in the IA, an additional $\log N$ bits are required. This information indicates to the CLIQ vector at the receiving side, which position is to be changed to non-empty. Hence, the CLIQ vector will only be written by arriving packets, whereas there at most two changes possible per tu.

The algorithm works as shown in Fig. 7.28 considering again six reception buffers which are shown in the top line. The winner according to a LOF strategy is highlighted in the top to demonstrate the differences to LOF-CLIQ. The vacancy levels are filtered (logical AND)

with the CLIQ vector, which is shown in the line underneath. The outcome is labelled CLIQ vacancies. If there is no selection after the filtering, the result of the LOF procedure is taken. If there are multiple winners, the selection pointer decides. In our example the filtering produces one result and reception buffer number **six** would be eligible to return its credit.

From an algorithmic point of view LOF is executed on asserted ('1') CLIQ positions. If this produces no result, LOF across the non-asserted ('0') CLIQ positions is taken instead. This procedure prioritizes the selection of credits corresponding to non-empty VOQs. The algorithm can be extended easily to non-binary CLIQ values, allowing to refine prioritization.

Temporal strategy

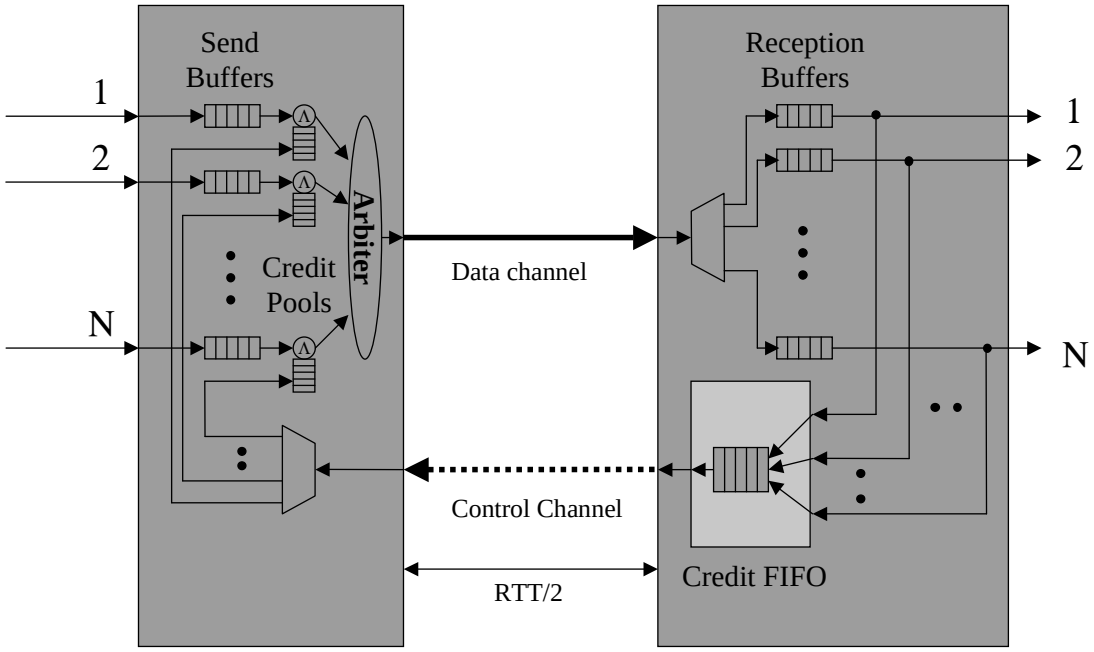


Figure 7.29: Temporal RXS (FIFO) Strategy

The temporal strategy selects the next credit to return based on preceding packet departures. This is displayed in Fig. 7.29. This strategy is realized by a FIFO, whereby the writing of the credits to the FIFO in the case of parallel departures is random. The temporal order is maintained only at a packet cycle granularity. From switch-state perspective, this strategy is again random, and therefore we expect it to behave similar to the RR scheme.

7.6 Performance Results

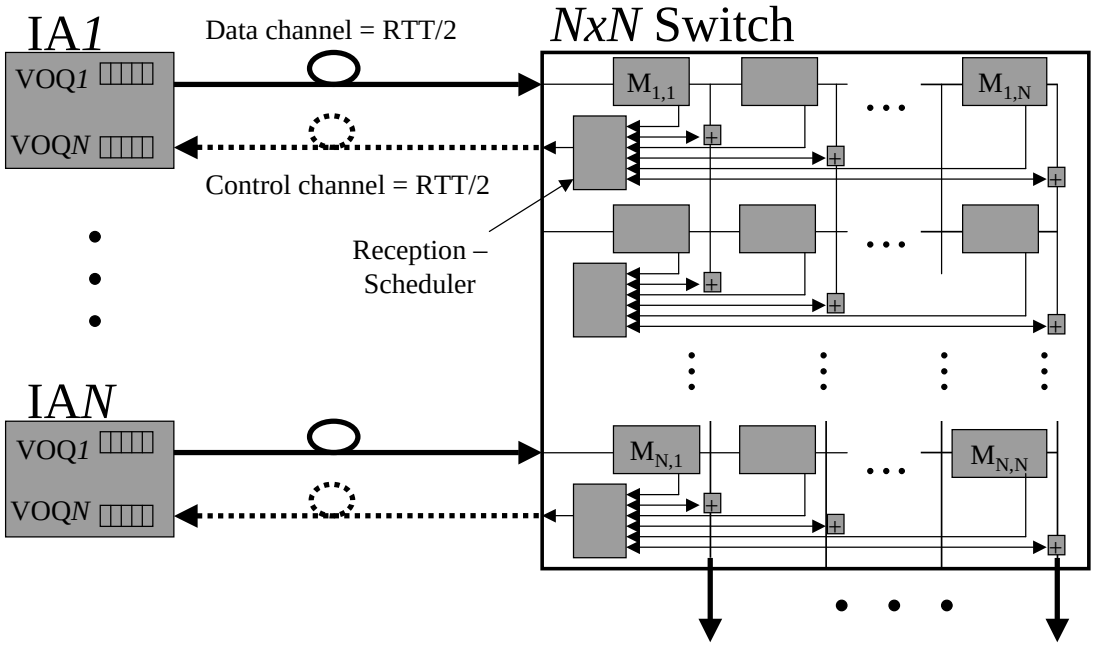


Figure 7.30: Exemplary CICQ Switch Architecture Using N RXS Modules

In Fig. 7.30 we see a CICQ switch equipped with N RXS modules, one per FCD. Each RXS runs independently from the others. This module may implement strategies of returning credits based on random, temporal, or spatial properties as outlined earlier. The number of credits to be returned is counted in a separate counter per FCSD, as indicated by the boxes with the "+" sign. This counter performs the bookkeeping of packet departures, e.g. add operation for HOF strategy and credit departures, e.g. subtract operation for HOF strategy.

We operate our switching system, shown in Fig. 7.30, at saturation level (input load of $L = 98\%$), which is best suited to reveal the different scheduling behaviors. We use an RTT of 64, which is our *design point* . A cross-point memory size of $M = 64$ is assumed. For the FIFO return strategy the model presented in Fig. 7.9 is taken. The spatial strategies that we study are based on buffer occupancy level, which in Fig. 7.30 is represented by the arrows going from memories $M_{i,j}$ to a reception scheduler and back. This information is not relevant when using the RR return strategy. We use the terms buffers and memories interchangeably, as there is one memory dedicated to each reception buffer. The memory must store all packets targeting a given reception buffer.

In the following performance analyses we study the behavior of the five different RXS strategies, namely FIFO, HOF, LOF, LOF-CLIQ, and RR under different scenarios. We always compare it to the Reference, which is a FC channel that is unscheduled and is capable of returning up to N FC events. All RXS strategies are restricted to return just one FC event per tu and hence FC scheduling is applied.

For scalability reasons we are interested in the impact of system size on proposed strategies. First of all we want to see how the different strategies compare to the Reference case using our default system parameters. We are pursuing the question of which strategy is able to exploit the phenomenon of credit contention and why. This analysis is performed in Subsec-

tion 7.6.1. As memory is expensive in terms of chip area, we also want to examine the impact of memory size on performance. A memory size of RTT packet locations is a necessity for work-conservation per FCSD. However, from a global perspective, this amount of memory appears to be excessive. This analysis is important with regard to efficiently supporting QoS requirements, although we do not study it here. For instance, if only a fraction of $M = 100\%\,\mathrm{RTT}$ is necessary to deliver good performance, then there is more space left to support QoS. Therefore, we study the influence of this parameter on the system latency, and study memory sizes between $M = 20\%\,\mathrm{RTT}$ and $200\%\,\mathrm{RTT}$. This investigation is conducted in Subsection 7.6.2. A reasonable assumption of the RTT of future high-speed switches is 64[41]. However, for today's high-speed switches this seems rather large. We therefore want to evaluate the impact of link length on performance. These experimentations are carried out in Subsection 7.6.3. As traffic may not always be uniformly distributed in destinations, we are interested in the impact of traffic with non-uniform spatial distributed destinations. A non-uniform traffic behavior is for instance encountered in the workload of MP systems. The results obtained are presented in Subsection 7.6.4. As traffic encountered in MP environments is very specific in terms of destination distribution and temporal variation, we check the scheduling strategies against various workloads taken from the SPLASH-2 suite. The experiments and results are outlined in Subsection 7.6.5.

7.6.1 Comparing Strategies under Design Point Assumptions

Motivation

There is a trend to make scheduling on the data-path more complex. This is to satisfy the different QoS needs and to improve performance. Building ever more complex schedulers is costly in terms of chip area and processing time. The problem of complex schedulers that achieve better performance is that they needed full state information of the entire FCD (distributed scheduling [96]) or of the entire switch (centralized scheduling [91]), which is again costly in terms of bandwidth and against our goal of diminishing FC bandwidth. Assuming a buffered switch with distributed scheduling, full state information of the entire FCD would not be required if the data-path scheduler would be supported from the reception side, i.e. scheduling was distributed within a FCD. The two immediate advantages are:

- savings in FC bandwidth
- simpler data-path scheduler

This implies that the switch is able to *pull* traffic from the send queues to achieve a better load balancing and hence better performance. The order of conveying FC information (credits) could be used to fulfill this task. The only purpose here of the data-path scheduler is to provide fairness, which is achieved by a round-robin. At last, distributed scheduling within a FCD would allow the sending node to push low latency traffic under low loads, but give the opportunity to pull traffic under high loads. To the best of our knowledge, there is no proof so far that such a scheme would indeed efficiently work.

Goal

The goal is to quantify under design-point assumptions the differences in system *delay* when applying RXS strategies as introduced in Sec. 7.5 with only one FC event channel capacity in comparison to the Reference, with a capacity of N. We are pursuing the question whether FC

scheduling influences performance and if so under what circumstances to ultimately formulate robust design rules for an RXS algorithm.

Experimental setup

The system parameters with respect to RTT and memory size are $M = \mathrm{RTT} = 64$. We apply an input load of $L = 98\%$. The results are shown for the three relevant system sizes: $N = 8, 16$, and 32 displaying the behavior for changing burst sizes: BS=10, and 30.

Results

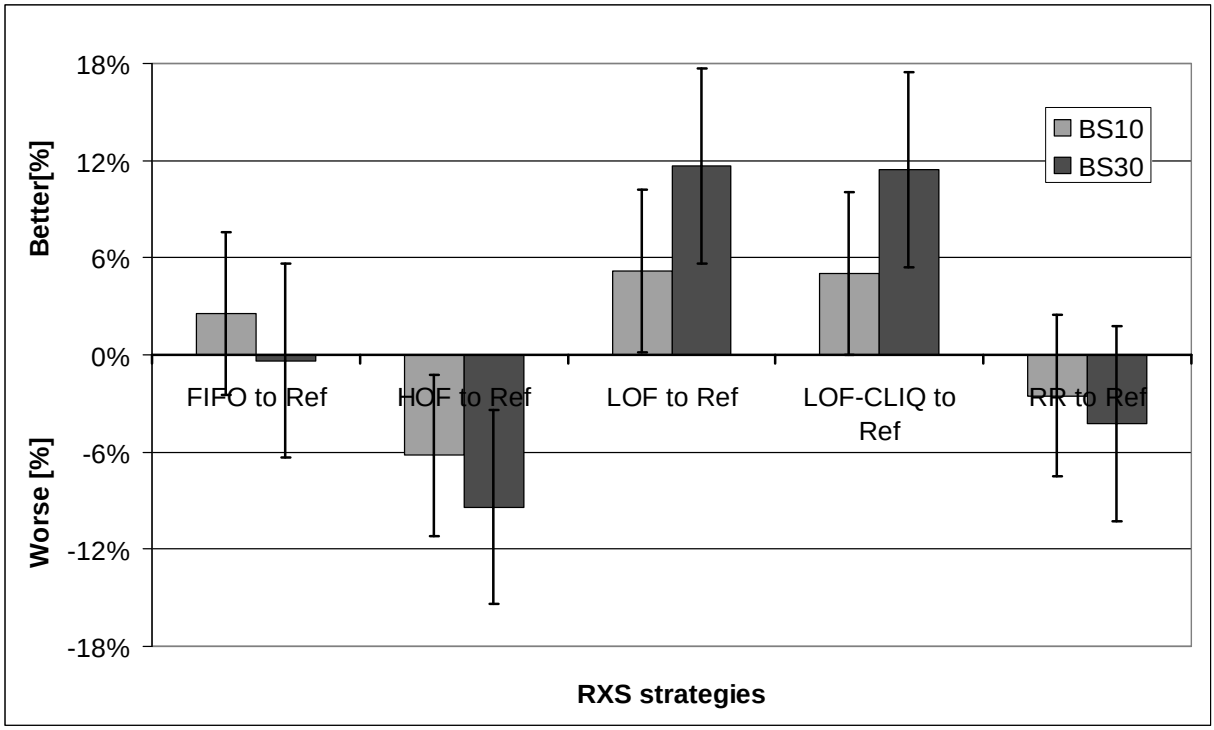


Figure 7.31: Strategies in Comparison to Unscheduled FC and Bandwidth of N (Reference): $N = 8$, L=98

The curves exhibited in Figs. 7.31 - 7.33 show on the y-axis by how many percent a strategy is better (positive scale) or worse (negative scale) than compared to the Reference. The metric of comparison is system delay and the ratio r displayed is calculated as

$$r = 100\frac{d_{ref} - d_x}{d_{ref}}, \tag{7.10}$$

whereby d_{ref} is the system delay for the Reference case and d_x the system delay of the RXS strategy under study. The error on system delay is smaller than 3% with a confidence of 90%. For non-overlapping confidence intervals the difference of the mean between the Reference and any strategy must be larger than 6% in order to be statistical significant. The (fixed) 6% confidence interval shown in Fig. 7.31 - 7.33 must not include zero for the RXS strategy to be significantly different than the Reference. For burst sizes of BS=10, as well as the combination of BS=30 and $N = 32$, 5% is used as the error obtained here is less than 2.5%.

Table 7.6 lists by how many percent the delay characteristic of the five different strategies differs to the Reference.

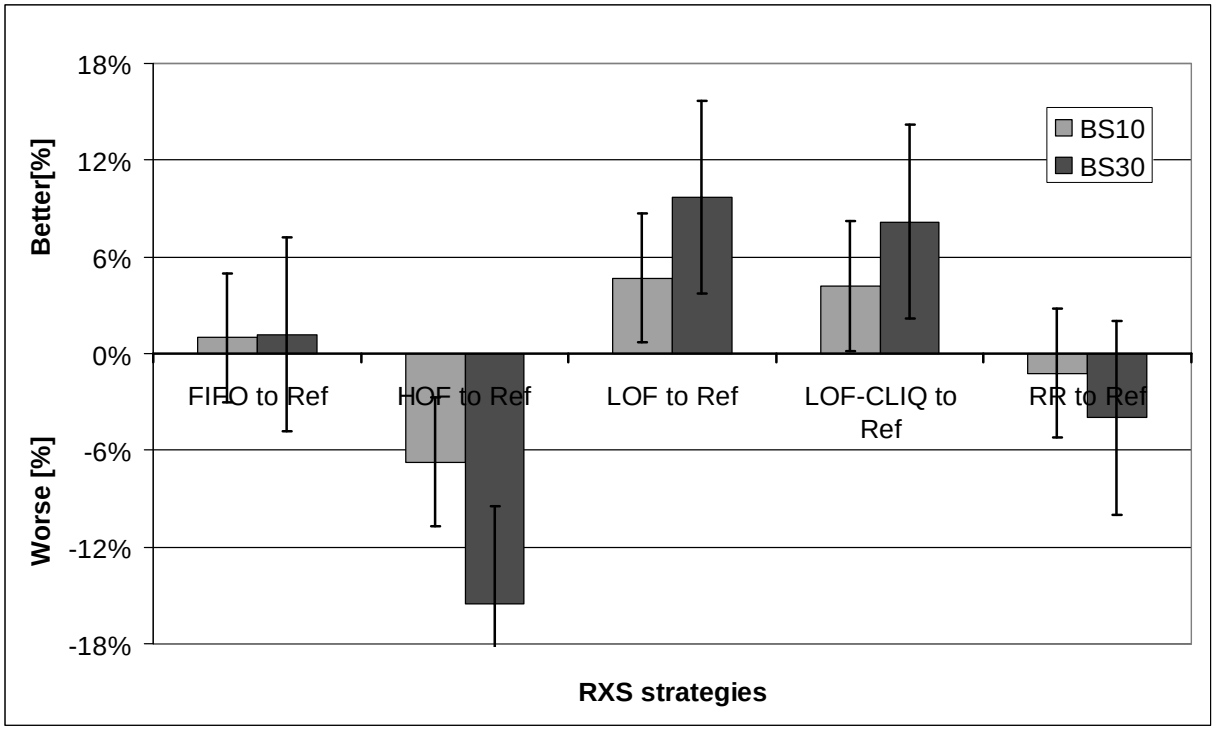


Figure 7.32: Strategies in Comparison to Unscheduled FC and Bandwidth of N (Reference): $N = 16$, L=98

All three graphs confirm that credit scheduling does have an impact on system delay. It is seen that the HOF strategy has a negative, while the LOF and LOF-CLIQ have a positive effect on system delay. For instance for $N = 8$ and BS=30 in Fig. 7.31, system delay for HOF is about 10% higher, while LOF and LOF-CLIQ achieve a 12% lower delay. There is no statistical difference inbetween the FIFO, the RR and the Reference across all system sizes, as zero is included in the confidence interval. For a burst size of BS=30 the results are more distinct than for a burst size of 10. For system sizes of $N = 8$, and 16, in Figs. 7.31 and 7.32, LOF and LOF-CLIQ are significantly different – better – than the Reference, while HOF performs significantly worse. For a system size of $N = 32$, the RXS makes a significant performance difference for $BS = 30$ and 60, but not for $BS = 10$. This is explained by a presumed interdependence of burst size and switch size. If the burst size is significantly less than the switch size $BS \ll N$, the performance impact is hardly visible. This is for instance the case for $N = 32$ and $BS = 10$. For $BS \approx N$ there is a significant difference already observed, for instance for $N = 8, 16$ and $BS = 10$, as well as $N = 32$ and $BS = 30$. For $BS > N$, as with $N = 8, 16$ and $BS = 30$, or

Table 7.6: Performance Gain/Loss of HOF, LOF, LOF-CLIQ, FIFO and RR Compared to Reference

	BS10			**BS30**			**BS60**
N	8	16	32	8	16	32	32
HOF	-6%	-7%	-3%	-9%	-15%	-7%	-8%
LOF	5%	5%	0%	12%	10%	5%	12%
LOF-CLIQ	5%	5%	3%	11%	8%	2%	-1%
FIFO	3%	1%	-2%	0%	1%	1%	-2%
RR	-3%	-1%	-1%	-4%	-4%	-2%	-1%

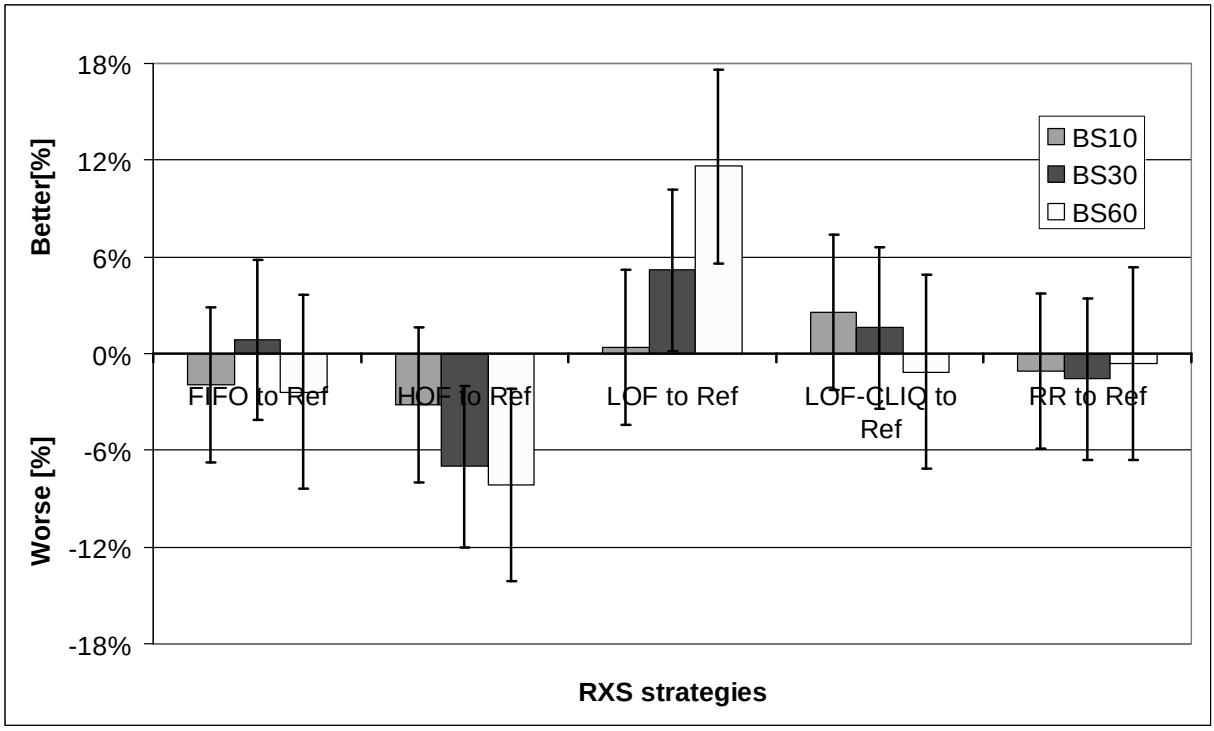


Figure 7.33: Strategies in Comparison to Unscheduled FC and Bandwidth of N (Reference): $N = 32$, L=98

$N = 32$ and $BS = 60$, the difference is even more significant.

Conclusion

Our results confirm that credit scheduling influences switch performance under high loads. It means that instead of making the data-path scheduler ever more complex to achieve better performance results, it suffices to leave it simple (RR) and distribute the decision making process across sender and receiver. The switch is best to decide which packets it want to receive in the near future in order to maintain a certain performance. It has the knowledge of the reception buffer states and can try to pull traffic to those reception buffers that are for instance more empty than others (LOF strategy). This causes a more equal distribution of packets stored at the reception side, a lower memory utilization of cross-point memories and consequently a lower delay, i.e. load-balancing is achieved.

The quality of the return decision – and consequently performance – should improve, if on top of LOF the state of the input queues is considered, because it is not worth returning credits to empty send queues at high priority. This is achieved with LOF-CLIQ and the results are expected to be equally good or superior to LOF. In Figs. 7.31 and 7.32 this is case, unlike Fig. 7.33, where the results of a system size of $N = 32$ are unveiled. The worse behavior of LOF-CLIQ as compared to LOF in the latter case is mainly explained by the large system size and the traffic pattern of uniform destinations, where ceased flows restart within short periods of time. With $N = 32$, the presence of enabled CLIQs of competing outputs is very likely, which delays the return of credits to empty send queues more than in cases of $N = 16$, or $N = 8$. Therefore, feasibility for a flow restart is more delayed, which is causing the worse behavior. For larger burst sizes this becomes even more severe, because a delayed restart is even more penalized. Such a case would call for better CLIQ tuning, where the send queue lengths are communicated at finer granularity, for instance empty, almost empty, and non-empty.

Nevertheless, LOF-CLIQ is in this particular case equally good as the Reference and a loss in performance is not observed.

The results also show that improper FC scheduling leads to a loss in performance: HOF is in most cases significantly worse than the Reference and compared to LOF the performance differences can be as high as 25%. Therefore, the algorithm of the RXS scheduler plays an important role and its design criteria must properly be defined.

7.6.2 Impact of Memory Size

Motivation

Memory size is a precious resource on a switch. It scales quadratically with system size for CICQ-switches. Its size per cross-point is determined by RTT and number of traffic classes p, such that the total amount of memory calculates as [45]

$$M_{tot} = pN^2\mathrm{RTT}. \tag{7.11}$$

Traffic classes include for instance priorities, guaranteed services, or lanes, as already pointed out in Subsection 4.4. For an RTT of 64, $N = 32$ and $p = 8$ there is a total of$M_{tot} = 8{*}32^2{*}64 = 524288$ packet locations necessary. This equals to 32 MBytes of memory assuming 64 byte packets. This amount of memory is required, in order to be work-conserving in between arbitrary input-output pairs. This might not be a requirement from a traffic engineering stand point. Assume for instance a OC-768 link, which carries 4 OC-192 flows. In this case, only 25% of the credits of a OC-768 flow are necessary to maintain work-conservation per FCSD and OC-192 flow. If all 4 multiplexed flows would go to the same destination then only 25% of the required bandwidth could be offered. This would be the trade-off. We do not discuss whether this is worthwhile doing or not. It is a traffic engineers decision. We just want to study whether it would be possible and at what (performance) cost. Definitely, a lot of memory could be saved. For instance, if only a fraction x was necessary to deliver equal aggregate performance than the amount of memory could be reduced to $M' = xM_{tot}$. If an upper limit of memory on a switch chip was given, $x < 1$ gave more room to support more QoS needs, i.e. more traffic classes.

Goal

The goal is to demonstrate that M can be reduced without loss in performance under uniform traffic.

Experimental Setup

The switch studied is as shown in Figs. 7.30. We vary memory size for a default of $\mathrm{RTT} = 64$ from $M_{i,j} = 20\%\mathrm{RTT}$ to $200\%\mathrm{RTT}$. Simulations are stopped as soon as the error of system delay is less than 3% with a confidence of 90%. We present the results for system sizes of $N = 8, 16,$ and 32 and for burst sizes of BS=10, 30, and 60. Packet destinations are uniformly distributed.

Results

We will present the trends and observations that are in common for all switch and burst sizes evaluated first. This is done by means of Fig. 7.34. The detailed results of the different burst and system sizes are found in Figs. 7.35–7.37.

Fig. 7.34 displays system delay over cross-point memory sizes. Each data point shows the confidence interval. The lines interconnecting those points are just a means to better visualize the trend. The results are compared to the bold solid line, which represents the Reference.

In general, there is an order in which the different strategies – including the Reference – perform. The HOF strategy delivers the worst performance of all strategies investigated, followed by RR and the Reference. Better than these are FIFO, followed by LOF-CLIQ. The best results are achieved by the LOF strategy.

The RR strategy, the Reference and the FIFO strategy at most points are statistically not significantly different from each other. Both curves are always positioned between HOF, LOF, and LOF-CLIQ. This can be verified in all figures. This confirms our initial assumptions on the random and temporal strategies.

Independent of fabric and memory size, LOF outperforms all other strategies, owing to its load-balancing capabilities. For the good delay that the LOF strategy achieves at $M = 100\%$ RTT, the Reference requires roughly twice the memory size.

System delay (Figs. 7.35–7.37) clearly confirms that we indeed operate at the saturation point for all results, because system delay is significantly larger than the minimum latency of 34 packet cycles (1 each per IA and switch, and 32 on the data channel).

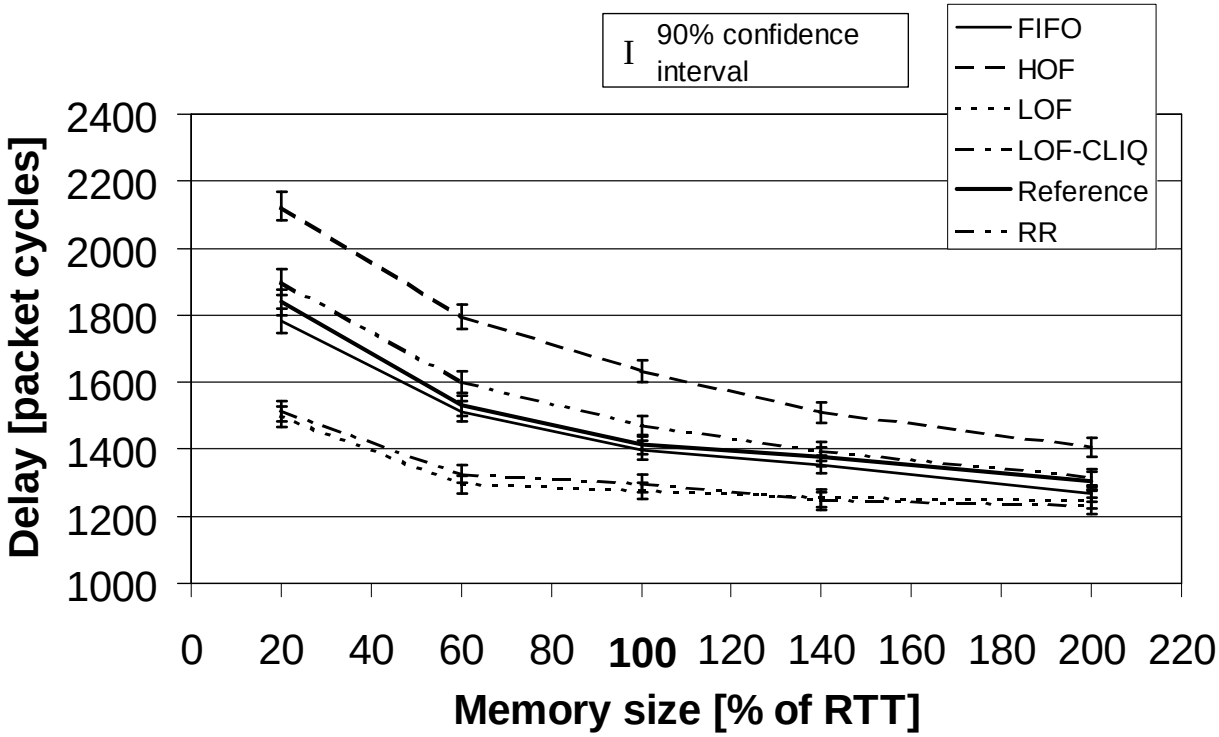


Figure 7.34: Impact of Buffer Size on Performance, Exemplary Result: $N = 16$, BS=30

Independent of fabric size, it is observed that system latency depends on memory size in the following manner: There is a steep drop in latency from a memory size of 20% up to 60% RTT. From this point on, the LOF strategy levels out, whereas the other strategies are slowly converging. This effect is weaker for BS = 10, but much stronger for BS = 30 and 60. As a consequence of a performance/memory tradeoff, an optimum memory size would be found using the LOF strategy at around 60% RTT's worth of memory. This result supports the initial assumption that an RTT's worth of reception buffer size is not a performance necessity, but that it rather is a requirement for being work-conserving in the absence of output contention. If the traffic distribution were not uniform, larger memories would be required to achieve the same performance. Still, having identified this knee gives more leeway for the implementation of QoS requirements. This is an important result, as the total amount of buffer space feasible on a switch is limited.

For Figs. 7.35–7.37 we observe that the minimal absolute system delay grows as the number of ports is increased. This is explained by the larger amount of buffer for larger switch sizes. At saturation, these buffers fill and henceforth the average waiting time for a packet in the switch is also larger, which has an impact on overall system delay.

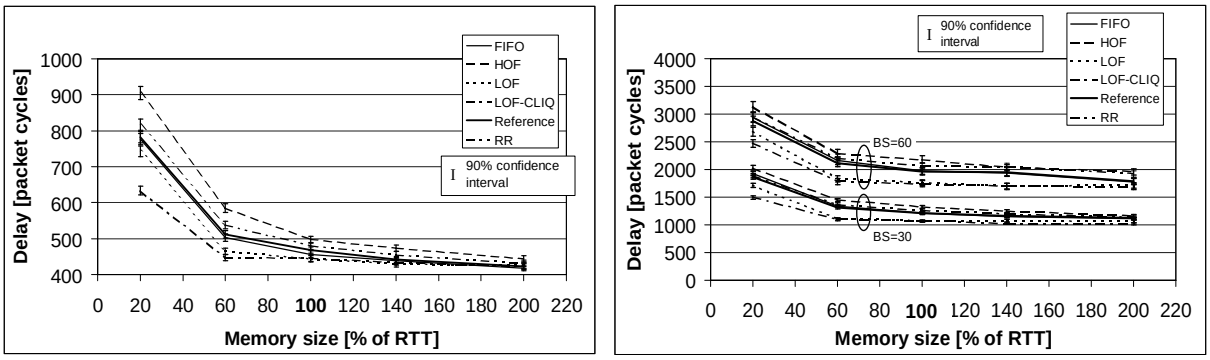


Figure 7.35: Impact of Buffer Size on Performance $N = 8$; (a) BS=10, (b) BS=30,60

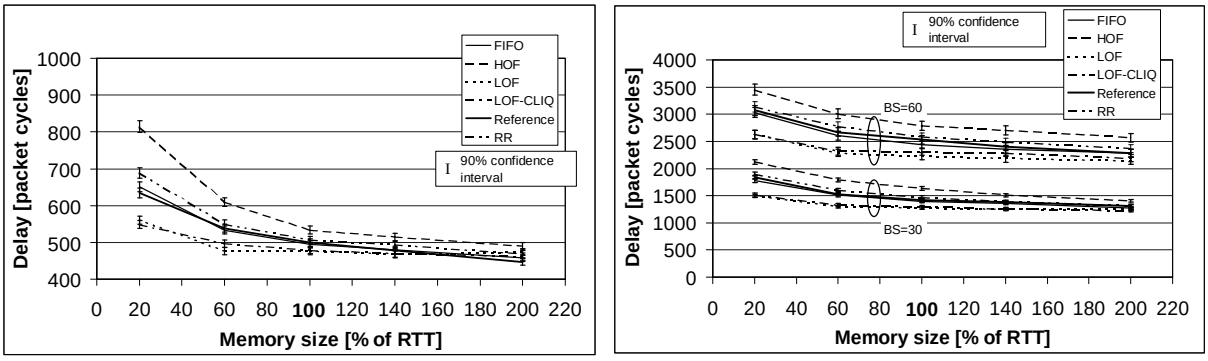


Figure 7.36: Impact of Buffer Size on Performance $N = 16$; (a) BS=10, (b) BS=30, 60

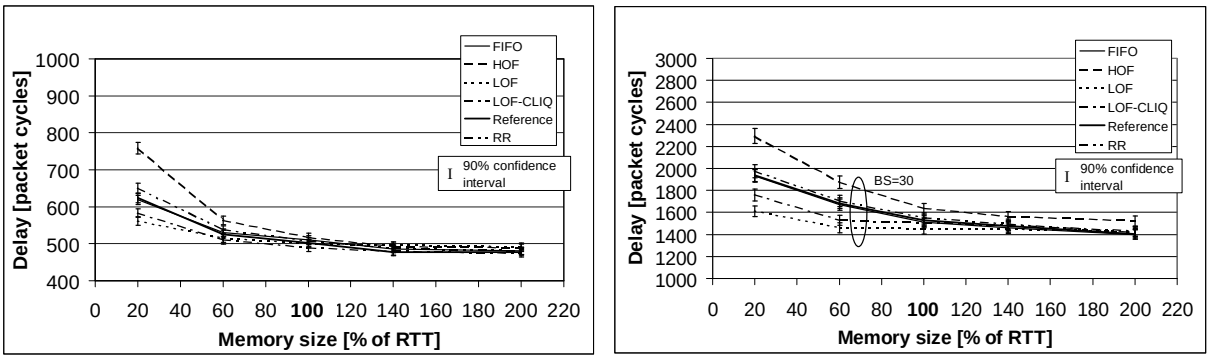


Figure 7.37: Impact of Buffer Size on Performance $N = 32$; (a) BS=10, (b) BS=30

The points on the curves of Figs. 7.35 to 7.37 that are most interesting to us are the one for $M = 60\%$ and 100% RTT. The latter, because it is the design point . The former, because it is an apparent sweet spot. The performance differences in comparison to the Reference that were achieved by the five RXS strategies for variable burst sizes and memory sizes of 60, and 100% are listed in Table 7.7. The numbers are in percent, whereby positive values disclose better system delay than the Reference and negative values worse delay.

We want to summarize the data by statistical analysis. We do a two factor full factorial design, whereas one factor is the RXS strategy, the other one system size. We compute the effects to determine the importance of the factors and analyze the variance using an F-test to statistically test the significance. We use an confidence level of 99%. The technique is explained in more detail in Subsection 7.6.4. The importance and significance of the two factors using this approach are summarized in Table 7.8. We see that a RXS is an important parameter in switch design as at least 76% of the total variation is explained by it. It also is a significant factor, whereas the number of ports is not. This is validated throughout all parameter, i.e. burst and

Table 7.7: Performance Differences (in %) of RXS Strategies in Comparison to Reference for Variation in Memory M and Burst Size BS, and Ports N

		M=60%			**M=100%**		
Burst Size	RXS	$N=8$	$N=16$	$N=32$	$N=8$	$N=16$	$N=32$
BS=10	FIFO	2	1	-1	3	1	-2
	HOF	-14	-13	-7	-6	-7	-3
	LOF	9	11	2	5	5	0
	LOF-CLIQ	13	14	3	5	5	3
	RR	-5	-2	-2	3	-1	-1
BS=30	FIFO	-1	1	1	0	1	1
	HOF	-10	-15	-7	-9	-15	-7
	LOF	16	15	13	12	10	5
	LOF-CLIQ	17	13	9	11	8	2
	RR	-4	-5	-1	-4	-4	-2
BS=60	FIFO	-2	3	—	-1	3	-2
	HOF	-9	-12	—	-11	-10	-8
	LOF	13	14	—	10	12	12
	LOF-CLIQ	15	13	—	11	9	-1
	RR	-4	-4	—	-5	-2	-1

Table 7.8: Importance and Significance of the two Factors RXS and System Size N

	Variation		**significant ?**	
BS/M	**Ports**	RXS	**Ports**	RXS
10/60%	2.5%	84.6%	no	yes
10/100%	8.0%	76.3%	no	yes
30/60%	0.6%	95.6%	no	yes
30/100%	1.7%	87.5%	no	yes
60/60%	0.0%	98.0%	no	yes
60/100%	0.9%	86.7%	no	yes

memory size combinations that we simulated for.

This result allows us to consider the results to be independent of system size. The mean performance differences achieved, independent of switch size are therefore shown in Table 7.9. We see that FIFO achieves system delays that are close to the Reference case. RR is about 4% worse than FIFO. HOF achieves results that are about 6 to 13% worse on average than the Reference. The best results are achieved by LOF and LOF-CLIQ. They achieve up to 12% better results for these memory sizes. We further want to summarize the two best strategies and determine the median and the index of dispersion for the values of Table 7.9. A more positive median and a smaller index of dispersion is better. The latter is found using the Semi-Inter Quartile-Range (SIQR). More details on the procedure of the calculation are presented in Subsection 7.6.4. This calculation delivers the following results

	LOF	LOF-CLIQ
SIQR	2.1	2.35
Median	8.4	6.4

LOF has a SIQR of 2.1% dispersion and a median of 8.4% of being better than the Reference.

Table 7.9: Performance Differences (in % and Averaged over Ports) of RXS Strategies in Comparison to Reference

	$BS = 10$		$BS = 30$		$BS = 60$	
M	**60%**	**100%**	**60%**	**100%**	**60%**	**100%**
FIFO	-0.1	0.1	-2.5	0.1	-2.2	-0.9
HOF	-12.1	-5.9	-13.5	-10.9	-13.2	-10.5
LOF	6.6	2.75	11.9	8.4	10.8	10.5
LOF-CLIQ	9.3	3.4	10.2	6.4	11.3	5.5
RR	-3.7	-0.3	-6.1	-3.9	-6.7	-4.5

Positive numbers reveal a better average system delay than measured for the Reference. Accordingly, negative numbers are worse than the Reference.

For LOF-CLIQ, these values are slightly worse with a median of 6.4 and a SIQR of 2.3%. We see that the LOF is superior to LOF-CLIQ, however the differences between these two strategies is narrow. Both achieve much better performance than the Reference.

Conclusion

Concerning the average memory occupancy when applying HOF, it is found that it is much higher than for LOF. This is the reason that system latency is so much greater for HOF. The higher memory occupancy of HOF is explained by that it supports the scheduling of packets to already filled reception buffers (or at least more filled than the others in the corresponding FCD). Assume, for instance, a FCD that is evenly filled with packets at its reception side and that credits contend on the reverse path. We describe the behavior when packets of one reception buffer make forward progress. As long as there are other credits to be returned, the current forward progress will not be seen until it is the only buffer that has credits to return. A pathological case may easily be constructed when the memory is full and is then emptied at a certain rate. There is a good chance that all credits will be tied up until they are returned. A new flow started during this time will not be served immediately. This is in contradiction to our initial expectation of achieving less credit underflow.

We can say that HOF favors memories that are almost full owing to strong contention at the output switch port, which is not a good idea as it will contribute to congestion in practical (real) systems.

The system parameters under which these experiments were conducted are summarized in Tab. 7.10.

Table 7.10: System Parameters of the Evaluation of the Influence of Memory Size on Switch Performance

RTT	M[%RTT]	System Size		
		8×8	16×16	32×32
RTT=4	20			
	40			
	60			
	100			
	140			
	200			
RTT=64	20	$\times$	$\times$	$\times$
	40	$\times$	$\times$	$\times$
	60	$\times$	$\times$	$\times$
	100	$\times$	$\times$	$\times$
	140	$\times$	$\times$	$\times$
	200	$\times$	$\times$	$\times$

7.6.3 Impact of Link Length

Motivation

Up to now, we performed our analyses for *long links* as we used $\mathrm{RTT} = 64$. This is a reasonable and justified assumption for future high-speed communication switches with a target aggregate bandwidth of 1 Tb/s and above (see Chap. 2). However, for switches that require less aggregate throughput, RTT may be significantly lower. Examples may be low-end communication switches, multiprocessor switches, or switches used in LAN/SAN environments. For this reason, it is interesting to know the FC scheduling behavior with small RTTs, i.e. *short links*.

Furthermore, for the prediction of the behavior of reception scheduling with increasingly larger RTTs – by a regression model for instance – at least two data points for different RTTs are necessary.

Goal

We want to find out how FC scheduling influences performance using short links. We are pursuing the question whether the influence is larger, smaller, or equal than what we have achieved using long links.

Experimental Set-Up

We compare our (long link) design-point ($M = \mathrm{RTT} = 64$) with a short link configuration of $M = \mathrm{RTT} = 4$. In both cases the memory size that offers work-conservation of arbitrary input-output flows.

The resulting total memory of the entire switch using short links with RTT=4 is significantly less as compared to long links with RTT=64 according to Eq. 7.11 with $p = 1$. This difference in total memory will have an impact on overall switch performance, which is the reason why we can not directly compare the performance results obtained. Instead, we have to compare the performance differences that our five RXS strategies achieve with respect to the Reference. For the system size we choose $N = 16$ and therefore do not repeat the entirety of the experiments conducted so far. The system is exercised with uniform traffic.

Results

Figures 7.38 to 7.41 show the system delay over throughput for input loads in between 90 and 98%. While Figures 7.38 and 7.39 display the results for a burst size of 10, Figs. 7.40 and 7.41 reveal the results for burst sizes of 30, and 60.

We see that the throughput for the system using long links is higher. This becomes apparent when comparing Figs.7.40 and 7.41. Here, the data points obtained for an input load of 98% of more shifted to the left in the upper figure (RTT=4) then in the lower one (RTT=64). This observation is explained by the larger memory for larger RTTs.

For low loads (up to approximately 90% input load) the mean system delay is lower for short than for long links. As we show only the fraction for high loads of the delay/throughput curve, this can only be seen when comparing the 90% data points for a burst size of 10.

It appear that the differences in the scheduling strategies are more distinct for short links. We will have to analyze this more closely in the following.

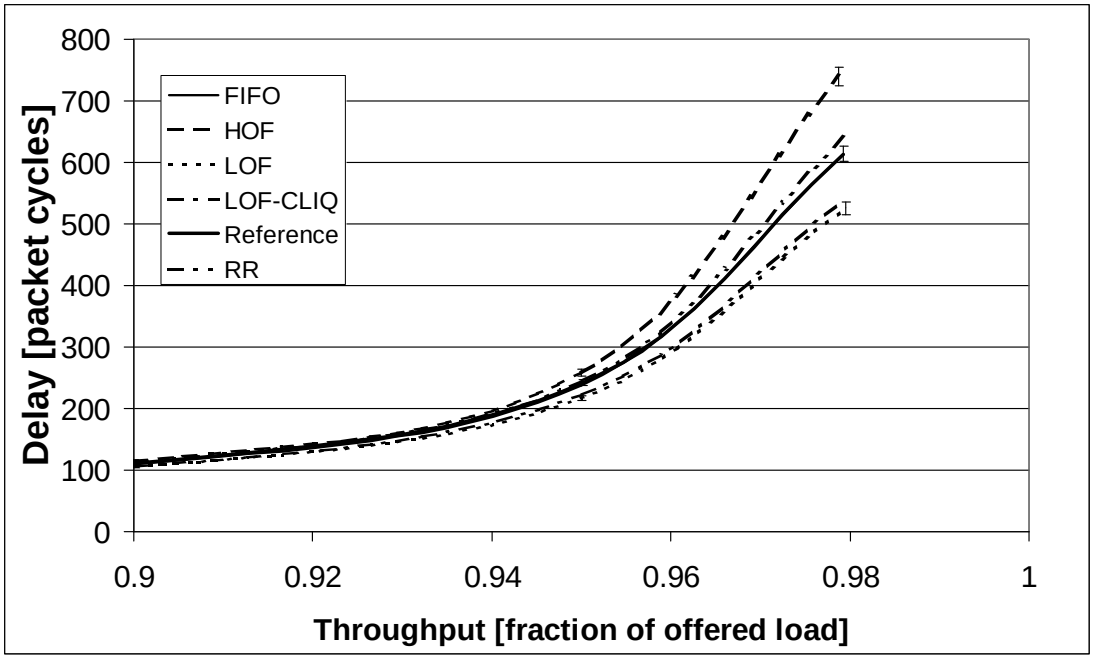


Figure 7.38: Switch Performance for High Loads, Small Bursts (BS=10), and Short Links (RTT=4), $N = 16$

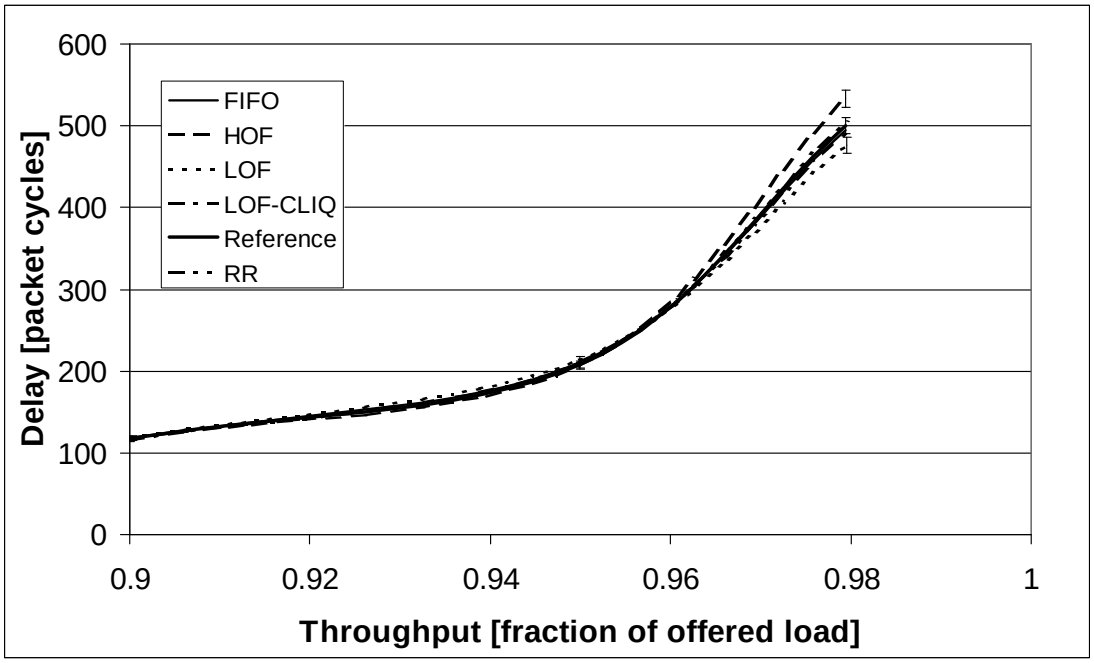


Figure 7.39: Switch Performance for High Loads, Small Bursts (BS=10), and Long Links (RTT=64), $N = 16$

Summarizing Data

The performance difference with respect to the Reference are listed in Table 7.11 for the traffic parameters input load and burst size. The numbers are in percent, whereby negative numbers signify a worse, positive numbers a better system delay. These percentages were calculated according to Eq. 7.10.

In order to give the results a statistical meaning, we do the following. As we conduct the

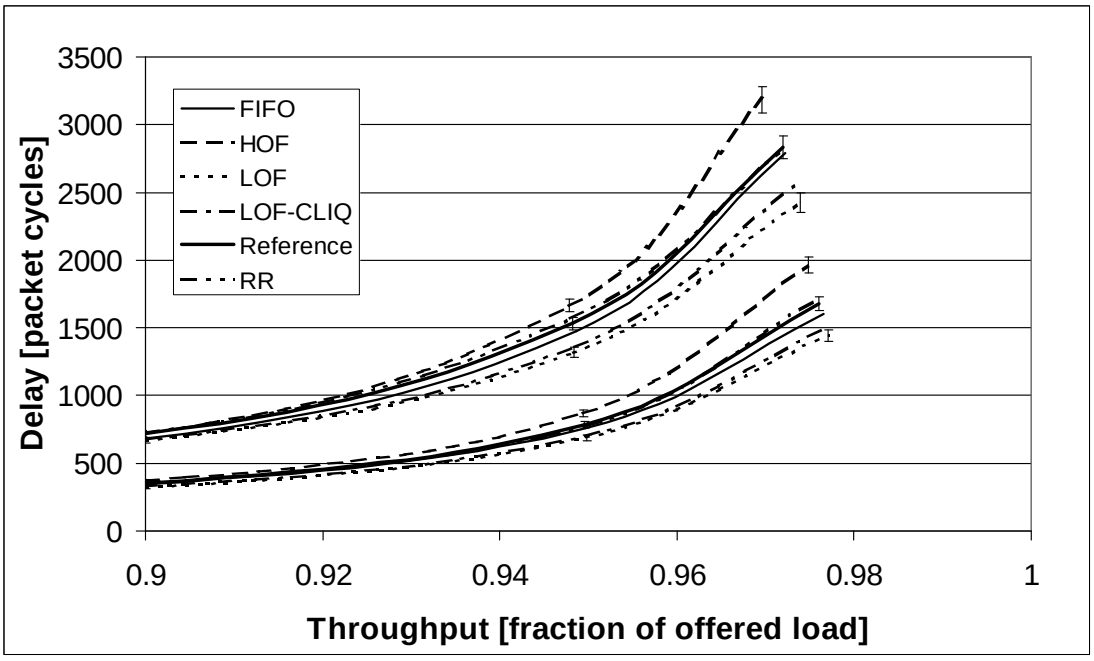


Figure 7.40: Switch Performance for High Loads, Large Bursts (BS=30, 60), and Short Links (RTT=4), $N = 16$

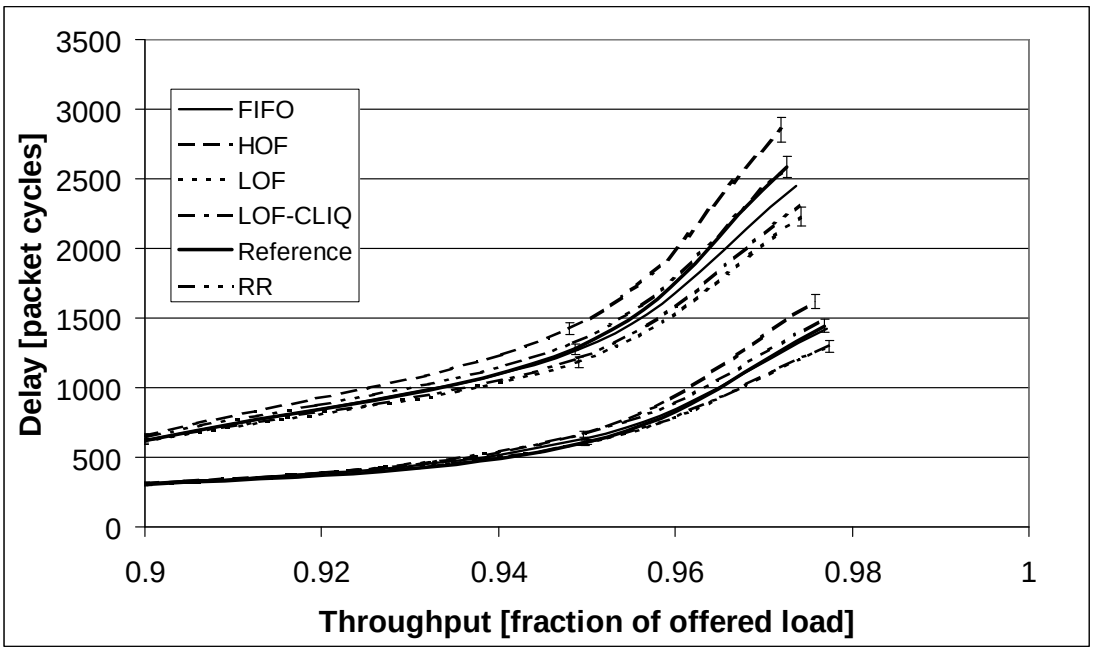


Figure 7.41: Switch Performance for High Loads, Large Bursts (BS=30, 60), and Long Links (RTT=64), $N = 16$

experiments on each of the two systems, one with short and the other one using long links, there is a one-to-one correspondence between the ith test on system A and the ith test on system B. We can therefore use analysis of paired observations. Here it suffices to compute the difference in performance gain or loss between the two systems and to determine the confidence interval. If it includes zero, the two systems are not significantly different from each other. If there is a significant difference, then the mean gives an indication whether the influence got larger or smaller.

Table 7.11: Performance Differences of RXS Strategies in Comparison to Link Length (RTT), Load L, and Ports N

		RTT=4			**RTT=64**		
Input Load	RXS	$BS10$	$BS30$	$BS60$	$BS10$	$BS30$	$BS60$
L=90	FIFO	1	3	4	1	-3	-1
	HOF	-4	-6	-1	1	-2	-6
	LOF	5	9	6	0	1	1
	LOF-CLIQ	4	6	5	2	1	0
	RR	-3	-1	-2	2	0	-5
L=95	FIFO	2	4	4	-1	-3	1
	HOF	-7	-11	-9	1	-9	-12
	LOF	11	12	14	-3	2	7
	LOF-CLIQ	8	10	12	-1	1	5
	RR	-1	1	-3	-1	-9	-4
L=98	FIFO	0	4	2	-1	3	-2
	HOF	-21	-17	-12	-11	-10	-8
	LOF	14	14	14	10	12	12
	LOF-CLIQ	13	10	10	11	9	-1
	RR	-4	-2	0	-5	-2	-1

We undertake the performance comparison per RXS strategy. This allows us to state for which strategy the link length has a significant impact. We use the 0.95-quantile of a t-variate with 8 degrees of freedom for the calculation of the confidence intervals. Its value is 1.860. We do the calculation of the confidence intervals exemplarily on HOF.

The paired observations are here $\{(-4,1),(-6,-2),(-1,-6),(-7,1),(-11,-9),(-9,-12),(-21,-7),(-17,-12),(-12,-10)\}$ with the values taken from Tab. 7.11. The first element of each pair represents the result for short links, whereas the second element the one for long links. The performance differences of each pair constitute a sample of 9 observations {-5, -4, 5, -8, -2, 3, -14, -5, -2}, whereby each sample pair of this list is calculated by taking the difference of each pair. For instance, the first sample calculates: $-4-1=-5$. The sample mean yields -3.56, whereas the sample variance is 31.78. The 90% confidence interval is therefore $-3.56 \mp (1.860)(1.88) = (-7.05,-0.06)$. This confidence interval is also listed for all RXS strategies in Tab. 7.12. For HOF, it does not include zero. Therefore, the two systems are significantly different. The mean performance loss for the system using short links is -9.78, but only -6.22 for the long link alternative. Thus, short links make the negative impact of HOF on performance using uniform traffic even more severe. For strategies that achieve better performance, we compare the mean performance gain. The mean performance gains or losses are included in Table 7.12. From these values we can conclude which link length has more importance on the results. The confidence intervals however give us a means to state statistical significance of the results.

We see that only the RR strategy does not deliver significantly different performance results for the link length analyzed, as its confidence interval includes zero. For all other strategies a shorter link length has a significant impact. This is noted in Table 7.12 in the column “more distinct”, which names this link length that has a larger impact on performance. Thus, the influence of FIFO, HOF, LOF, and LOF-CLIQ on system delay is larger when using shorter links.

Table 7.12: Importance and Significance of the Factor Link Length

RXS	Confidence Interval	Average Performance Gain/Loss RTT=4	RTT=64	Result significant?	more distinct?
FIFO	[0.50, 4.61]	2.67	0.11	y	short
HOF	[-7.05, -0.06]	-9.78	-6.22	y	short
LOF	[4.40, 9.38]	11.0	4.11	y	short
LOF-CLIQ	[2.83, 8.06]	8.67	3.22	y	short
RR	[-1.94, 3.28]	-1.67	-2.33	n	—

Conclusion

Table 7.13: System Parameters of the Evaluation of the Influence of the Link Length

RTT	M[%RTT]	System Size 8×8	16×16	32×32
RTT=4	20 40 60 100 140 200		×	
RTT=64	20 40 60 100 140 200		×	

We have analyzed the system using short links. As "short" is a relative number, we have chosen and RTT of 4, which equals a link length of 2. We have compared the results with the ones obtained using long links, i.e. RTT=64. The entire system parameters that we used in these experiments are listed in Table 7.13. We have drawn the delay/throughput curves for input loads in between 90 and 98%. We saw that the influence of the RXS strategies FIFO, HOF, LOF, and LOF-CLIQ are larger for a SF using short links. This was proven by statistical analysis using paired observations.

It turns out that FC scheduling has a more significant impact for short links. This means it is of particular interest for the design of today's switches. FC scheduling therefore, is a topic that has been neglected in the past, but that should have been addressed much earlier due to its importance. The performance benefits achieved on average are 11% for the buffer size large enough to provide work-conservation per FCSD for high loads when using LOF as compared to only 4% for long links.

7.6.4 Impact of Non-Uniform Traffic

Motivation

The RXS strategies have been analyzed so far for traffic with uniformly distributed destinations. This traffic assumption reflects an ideal spatial behavior that may be encountered in large trunks in a communications environment. However, even here this assumption is less true as unpredictable IP-traffic becomes even more dominant. In particular, in multiprocessor environments this clearly does not hold true. Here, it often occurs that one port within a FCD is favored over others, yielding a non-uniform traffic distribution.

This motivates us to study the behavior of the five RXS strategies under these conditions.

Goal

We want to find out how our five RXS strategies perform under traffic with non-uniform destination distribution.

Experimental Set-up

We conduct our experiments for short and long links (RTT=4 and 64). We vary the non-uniformity by parameter w, which describes the unbalanced probability. An unbalanced probability of $w = 0.0$ creates traffic with uniform destinations, whereas $w = 1.0$ produces directed, fully unbalanced, traffic. In the latter case, the cut-through latency is measured. Furthermore, we analyze for two different burst sizes. In one case, $BS = 1$ is chosen, which is also known as Bernoulli traffic, and in the other case we have $BS = 10$. In the former case we equip the switch with a cross-point memory size of $M = 100\%$ RTT, while in the latter $M = 200\%$ RTT.

We model a system size of $N = 8$ and 32. There are 1,024,000 packet simulated for the 8×8 configuration yielding 16'000 packets per cross-point and 3,076,000 for the 32×32, which results in 3,000 packets per cross-point. This reduction in number of packets simulated is motivated by a significant reduction in simulation time. It is justified, because we have no or only a very small burst size. Therefore the absolute results will certainly differ, but the relative performance between the RXS scheduling schemes will already be visible. Owing to the small burst sizes, we stop simulations as soon as the error on system delay is less than 2% on a 90% confidence.

Results

While Figures 7.42 to 7.45 reveal the behavior for long links, Figures 7.46 to 7.49 show the results for short links. As the unbalanced probability has two important points for $w = 0.0$ (uniform traffic) and $w = 1.0$ (directed traffic), these points are explicitly named in theses graphs. Across all curves we observe the largest differences for an unbalanced probability w in between 0.4 and 0.8. We choose $w = 0.6$ and 0.8 for further analysis. For all curves, the *Reference delivers the best results*, which means it corresponds to the lowest delay curve in the figures in the range of $w = 0.2$ to $w = 0.8$. Our figures do not resolve the behavior for $w = 0.0$, which we had to treat separately. We'll not repeat it here, because a similar study has already been conducted in Subsection 7.6.1. For $w = 1.0$ all results are the same, as we are dealing with directed traffic, which has been chosen such that destinations are not overlapping, i.e. each IA sends to another destination. Therefore, there is no output contention and the cut-through latency is measured. This point is therefore not of further interest to us, but it is useful as verification.

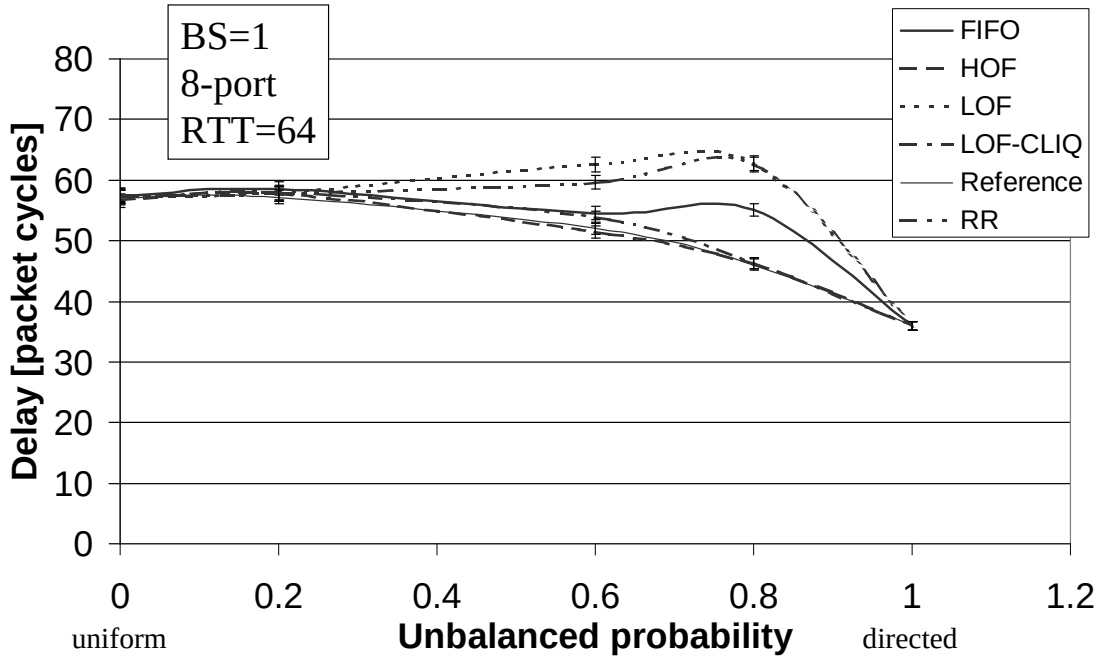


Figure 7.42: RXS Strategies Exposed to Non-Uniform, Non-Bursty (BS=1) Traffic, Long Links (RTT=64), $N = 8$, and $M = 100\%$ RTT $(M = 64)$

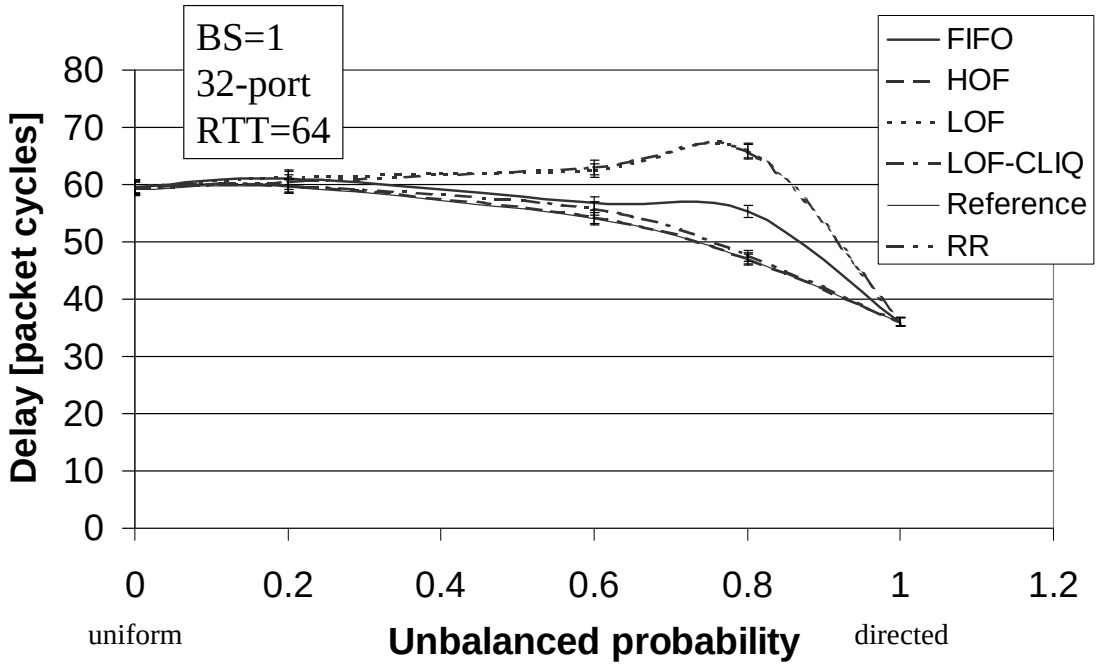


Figure 7.43: RXS Strategies Exposed to Non-Uniform, Non-Bursty (BS=1) Traffic, Long Links (RTT=64), $N = 32$, and $M = 100\%$ RTT $(M = 64)$

However, in contradiction to our observations from the case of uniform destinations, we see that the HOF strategy delivers better results than LOF. Here, the success/failure of the LOF and HOF strategies have inverted: LOF delivers the worst results now, whereas HOF comes very close to the Reference. The FIFO and RR strategies perform badly. Typically, RR is not significantly different than LOF, while FIFO is better. FIFO is found typically between LOF/RR and Reference/HOF/LOF-CLIQ.

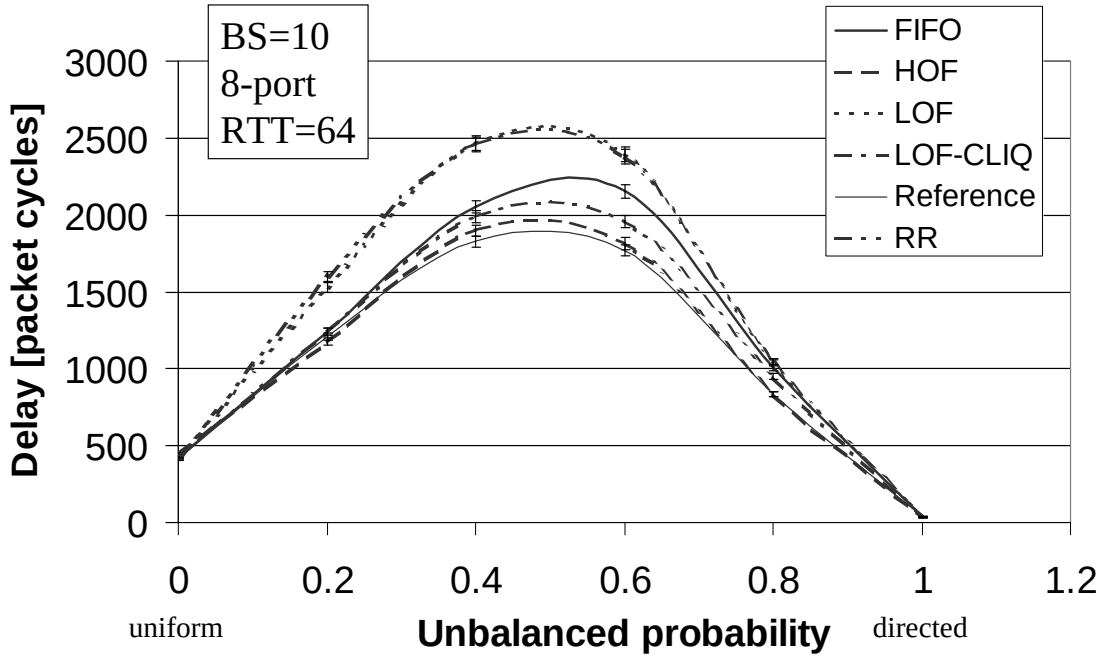


Figure 7.44: RXS Strategies Exposed to Non-Uniform, Bursty (BS=10) Traffic, Long Links (RTT=64), $N = 8$, and $M = 200\%\,\mathrm{RTT}\,(M = 128)$

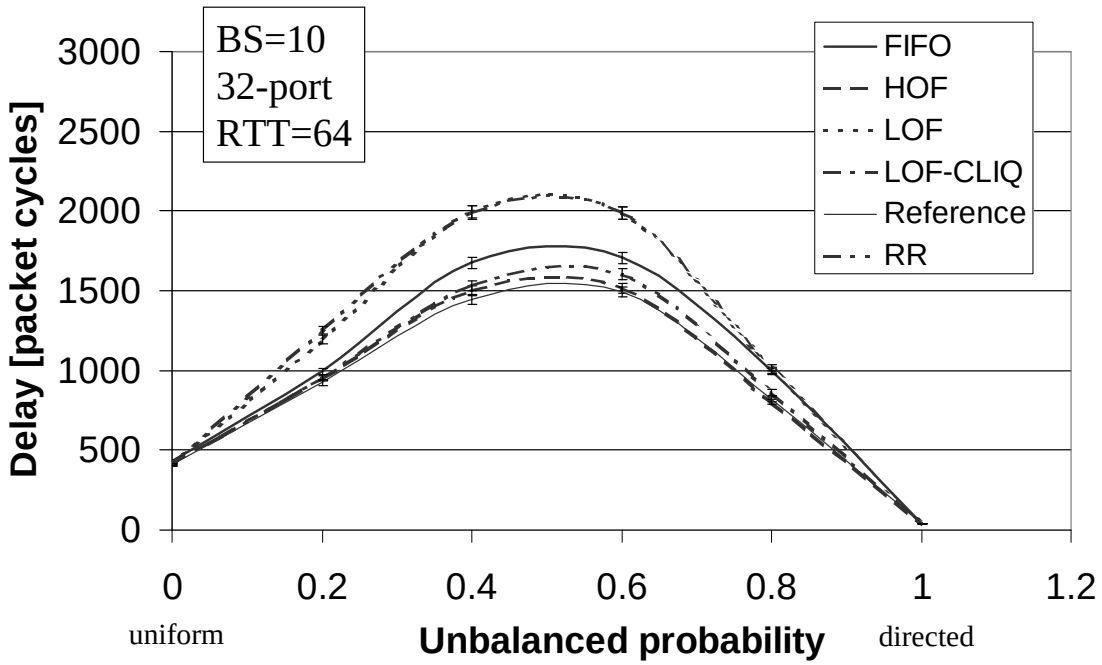


Figure 7.45: RXS Strategies Exposed to Non-Uniform, Bursty (BS=10) Traffic, Long Links (RTT=64), $N = 32$, and $M = 200\%\,\mathrm{RTT}\,(M = 128)$

The most important result is that – broadly speaking – the LOF-CLIQ strategy performs as well as HOF, and more importantly as well as the best case, the Reference. When looking at these two curves (HOF and LOF-CLIQ) in more detail, we make the following observation. Sometimes they are about equally good as for instance in Figs. 7.42 and 7.43, sometimes HOF is better than LOF-CLIQ as for instance in Figs. 7.44 and 7.45, or sometimes LOF-CLIQ outperforms HOF as in Figs. 7.46, 7.48 and 7.47, 7.49. We want to find out, which of the two

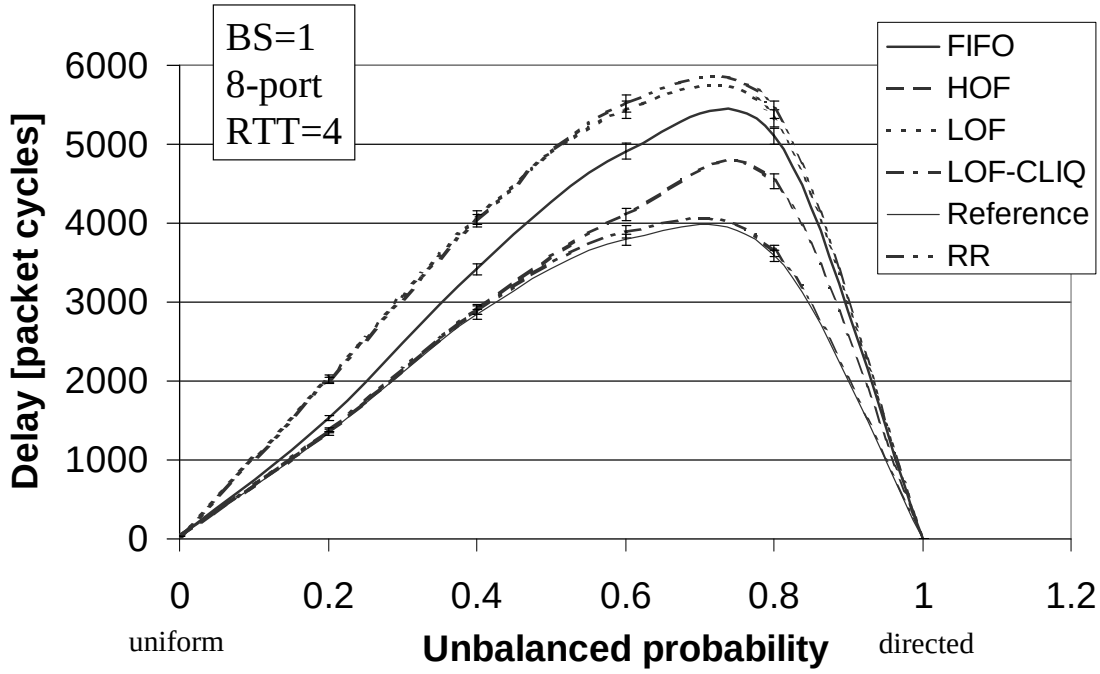


Figure 7.46: RXS Strategies Exposed to Non-Uniform, Non-Bursty (BS=1) Traffic, Short Links (RTT=4), $N = 8$, and $M = 100\%$ RTT $(M = 4)$

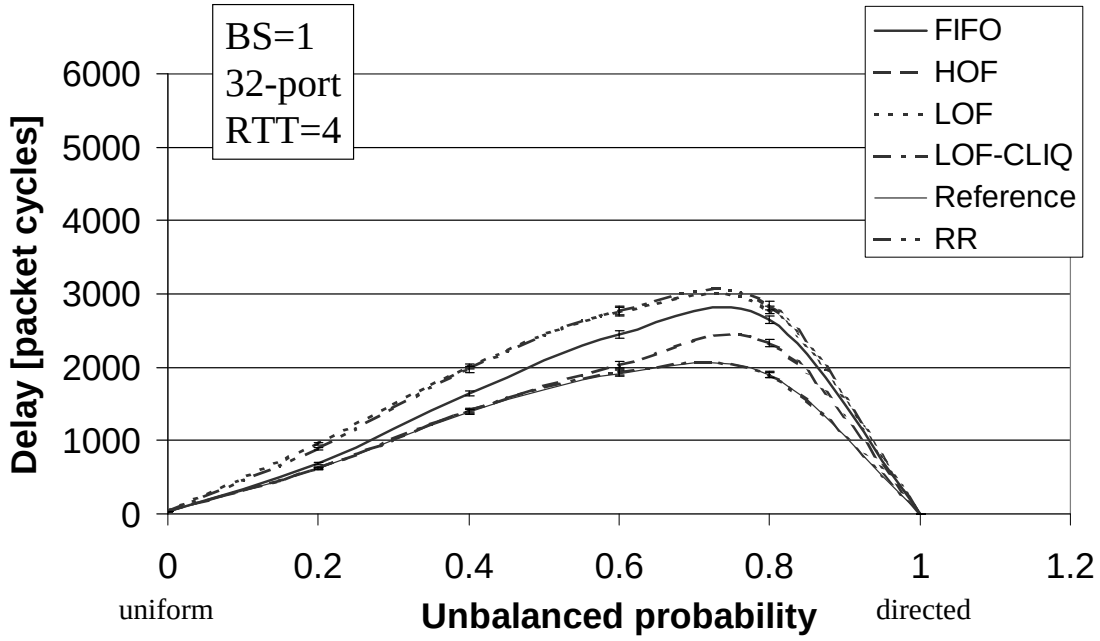


Figure 7.47: RXS Strategies Exposed to Non-Uniform, Non-Bursty (BS=1) Traffic, Short Links (RTT=4), $N = 32$, and $M = 100\%$ RTT $(M = 4)$

strategies can be considered better by statistical analysis. We therefore use the ratio r that describes the performance difference as introduced in Eq. 7.10. Negative values signify by how much percent a strategy performs worse than the Reference, while positive values indicate the percentage by which the Reference is outperformed. For equal strategies and unbalanced probabilities w, there is a certain variability of the delay observed. This can be verified in Table 7.14, which lists the values of r for the two RTTs (4 and 64) and burst sizes (1 and 10). Here, we see

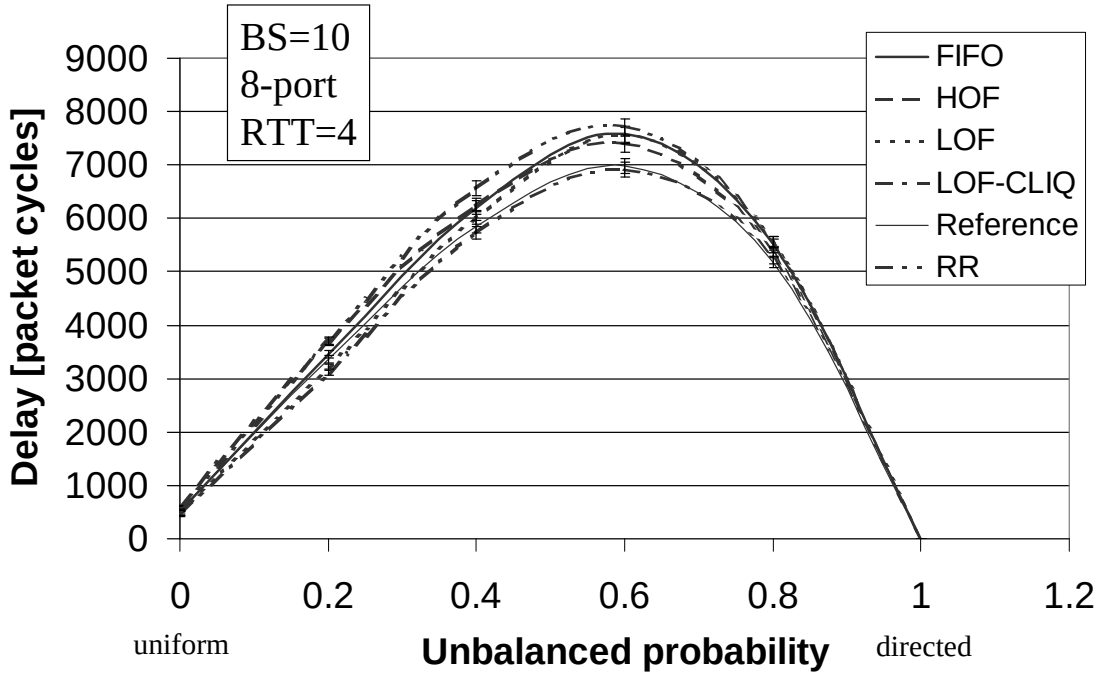


Figure 7.48: RXS Strategies Exposed to Non-Uniform, Bursty (BS=10) Traffic, Short Links (RTT=4), $N = 8$, and $M = 200\%$ RTT $(M = 8)$

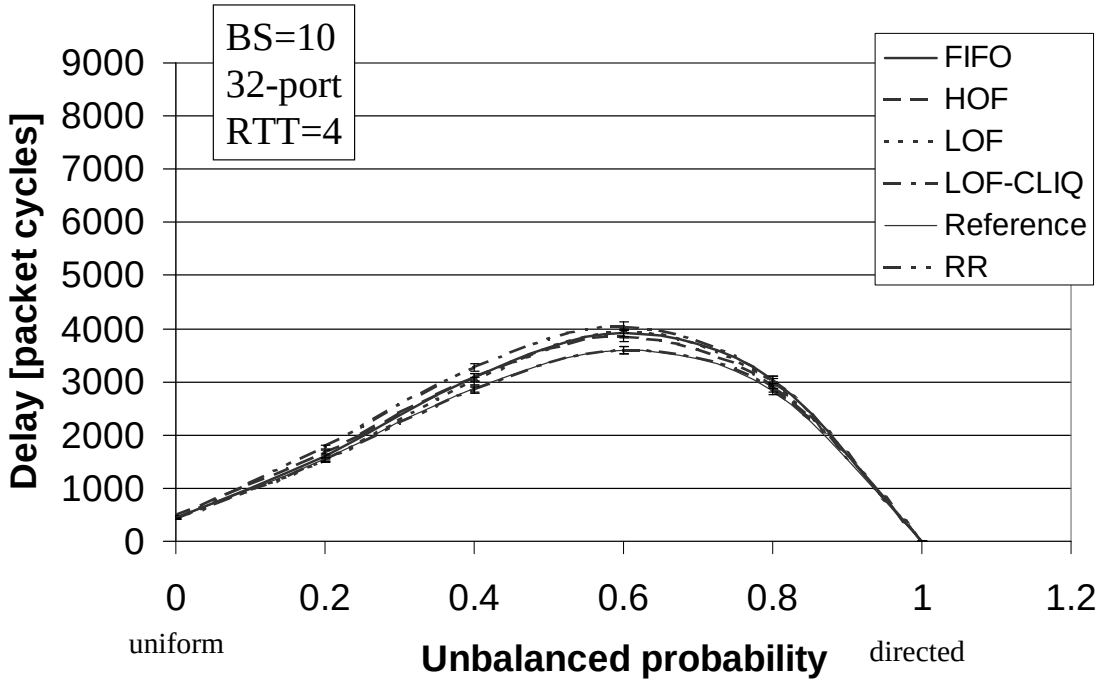


Figure 7.49: RXS Strategies Exposed to Non-Uniform, Bursty (BS=10) Traffic, Short Links (RTT=4), $N = 32$, and $M = 200\%$ RTT $(M = 8)$

for example that a RR RXS strategy can be up to 50% worse than the Reference.

As a criteria to determine the better strategy, we define that *this* strategy performs better whose variability (index of dispersion) is smaller for the parameters analyzed and whose ratio r (index of central tendency) is larger, as larger positive numbers are better.

Summarizing Data

Table 7.14: Performance Differences (in %) of RXS Strategies in Comparison to Reference for Variation in Degree of Non-uniformity of Traffic w, Burst Size BS, Link Length RTT, and Ports N

			$w = 0.6$		$w = 0.8$	
Burst Size	Link Length	RXS	$N = 8$	$N = 32$	$N = 8$	$N = 32$
BS=1	RTT=4	FIFO	-30	-28	-42	-40
		HOF	-8	-6	-26	-23
		LOF	-44	-44	-48	-47
		LOF-CLIQ	-3	-1	-2	0
		RR	-46	-45	-51	-50
	RTT=64	FIFO	-5	-5	-20	-18
		HOF	1	0	-1	0
		LOF	-20	-15	-36	-40
		LOF-CLIQ	-3	-3	-1	-1
		RR	-15	-16	-36	-40
BS=10	RTT=4	FIFO	-9	-9	-6	-8
		HOF	-6	-7	-4	-4
		LOF	-8	-9	-7	-6
		LOF-CLIQ	1	0	-1	-1
		RR	-11	-13	-7	-8
	RTT=64	FIFO	-22	-14	-21	-22
		HOF	-3	-1	0	2
		LOF	-35	-33	-24	-21
		LOF-CLIQ	-11	-7	-14	-5
		RR	-35	-33	-25	-24

We used the factors system size N, burst size BS, unbalanced probability w, RTT, and the RXS strategies to conduct the experiments that lead to the results shown in Figure 7.42 to 7.49. The ratio of performance difference to the Reference is listed in Table 7.14 for $w = 0.6$ and 0.8.

In a first step, we want to find out, whether there are factors that are neither important nor statistically significant. These factors could then be eliminated by taking their means. In a second step, we use the SIQR to determine the index of dispersion and the median to express the index of central tendency.

We note that the shape and therefore the differences in system delay are in about the same range for the two system sizes $N = 8$ and 32. We therefore do a two factor full factorial design [97] to study the impact on performance, whereas one factor is system size N and the other is the RXS strategy. We do such an analysis for each of the combinations RTT/BS/w, which results in eight independent calculations (see Table 7.17). Exemplarily, we execute this analysis for the results of the parameter combination RTT=64/BS=1/w=0.8. We calculate the effects, as done in Table 7.15.

This table shows that the mean difference of all strategies to the Reference is -19.3%. It is interpreted that on average the strategies perform 19.3% worse than the Reference. The effects say by how much percent a strategy or the system size deviates from this average. For instance the LOF-CLIQ performs only 1.0% (-19.3 + 18.3) worse.

In a next step we allocate the variation and analyze the variance. Both are shown in a tabular

Table 7.15: Calculation of the Effects for Non-uniform Traffic; $BS = 1$, RTT=64, $w = 0.8$

	$N = 8$	$N = 32$	**Sum**	**Mean**	**Effect**
FIFO	-20	-18	-38	-19	0.3
HOF	-1	0	-1	-0.5	18.8
LOF	-36	-40	-76	-38	-18.7
LOF-CLIQ	-1	-1	-2	-1	18.3
RR	-36	-40	-76	-38	-18.7
Sum:	-94	-99	—	—	—
Mean:	-18.8	-19.8	—	-19.3	—
Effect:	0.5	-0.5	—	—	—

arrangement of Table 7.16, which is also called Analysis Of Variance (ANOVA) table. The variation in this table explains by how much percent a single factor contributes to the total variation. The larger the variation explained by just one factor, the more important it is. Here, we see that the RXS strategies explain 99.34% of the total variation, while 0.0895% is attributed to the system size. The error calculated explains 0.5726% of the variation. We conclude that the choice of RXS strategy is an important parameter in the design of a FCD and the performance of the switch.

Table 7.16: ANOVA Table for Non-uniform Traffic; $BS = 1$, RTT=64, $w = 0.8$

Component	**SSQs**	**% Variation**	**DF**	**Mean Square**	**F-computed**	**F-table**
y	$SSY = 6519$					
$\overline{y}_{..}$	$SST = 2794$					
$y - \overline{y}_{..}$	$SS0 = 3735$	100.0	9			
Ports	$SSA = 2.5$	0.0895	1	2.5	0.625	21.2
RXS	$SSB = 2776$	99.34	4	693.9	173.5	15.98
Errors	$SSE = 16$	0.5726	4	4		

DF stands for degrees of freedom. y denotes the mean, whereas $\overline{y}_{..}$ the grand mean, and $y - \overline{y}_{..}$ the total variation. SSQ abbreviates the sum of squares.

After having determined the importance of the two factors we have to statistically test for significance. We therefore divide the sum of squares by their corresponding degrees of freedom, to get mean squares. We perform an F-test in Table 7.16, which checks whether "F-computed", the computed ratio of the individual mean squares to the means squares of error is greater than "F-table", the value read from the table of quantiles of F-variates (e.g. Tables A.6 to A.8 in [97]). If the calculated value is greater, the factor is assumed to explain a significant fraction of the variation. This test basically checks the contribution to the variation of the single factor with that of the errors. If a value is within the error, it is considered insignificant, otherwise it is significant. In Table 7.16 we see both values and see that the RXS strategy is a significant factor, whereas the number of ports has an insignificant influence. The F-test was performed at a 99% confidence level.

The results of the importance, which is the percentage of the variation a factor can explain, and the significance for the results of all 8 parameter combinations are summarized in Table 7.17. Indeed, the selection of a RXS strategy is an important and significant factor. These results empirically demonstrate that the influence of the number of ports on the performance re-

Table 7.17: Importance and Significance of the two Factors RXS and System Size N

RTT/BS/w	**Variation**		**Significance**	
	Ports	RXS	**Ports**	RXS
4/1/0.6	0.15%	99.8%	no	yes
4/1/0.8	0.31%	99.65%	no	yes
4/10/0.6	1.4%	98%	no	yes
4/10/0.8	0.65%	95.1%	no	yes
64/1/0.6	0.17%	97.4%	no	yes
64/1/0.8	0.09%	99.34%	no	yes
64/10/0.6	1.9%	97.33%	no	yes
64/10/0.8	2.0%	95.3%	no	yes

sults are insignificant and therefore the variation occurs independently of the system size of the switch. Owing to this empirical independence of system size, we summarize the performance differences by averaging over port size. This is shown in Table 7.18 for short links and 7.19 for long links.

Table 7.18: Performance Differences (in % and Averaged over Ports) of RXS Strategies in Comparison to Reference for Short Links (RTT=4)

	$BS = 1$		$BS = 10$	
	$w = 0.6$	$w = 0.8$	$w = 0.6$	$w = 0.8$
FIFO	-29	-41	-9	-7
HOF	-7	-24.5	-6.5	-4
LOF	-44	-47.5	-8.5	-6.5
LOF-CLIQ	-2	-1	0.5	-1
RR	-45	—	-12	-7.5

Table 7.19: Performance Differences (in %) of RXS Strategies in Comparison to Reference for Long Links (RTT=64)

	$BS = 1$		$BS = 10$	
	$w = 0.6$	$w = 0.8$	$w = 0.6$	$w = 0.8$
FIFO	-5	-19	-18	-21.5
HOF	0.5	-0.5	-2	1
LOF	-17.5	-38	-34	-22.5
LOF-CLIQ	-3	-1	-9	-9.5
RR	-15.5	-38	-34	-24.5

With these tables we have a better overview of the results. *We see that indeed HOF and LOF-CLIQ deliver the best results*. LOF-CLIQ is a good candidate in the case of short links. For long links HOF has advantages: similar to the discussion of Subsection 7.6.1, an improved CLIQ signalling would make LOF-CLIQ superior also in the case of long links. We now determine the SIQR and the median of these strategies, to find out which strategy actually performs better when considering systems with short and long RTTs. We turn our attention first to the SIQR to

express the variability of the results. For HOF the variability has a lower limit of -24.5% and an upper limit of 1%, whereas for LOF-CLIQ the boundaries are in between -9.5% and 0.5%. The ordered list of HOF taken from the averaged numbers of Table 7.18 for RTT of four and 7.19 for RTT of 64 is {-24.5, -7, -6.5, -4, -2, -0.5, 0.5, 1}, and for LOF-CLIQ it is {-9.5, -9, -3, -2, -1, -1, -1 , 0.5}. For the determination of the SIQR the 25%, or the first quartile Q_1, and the 75%, or third quartile Q_3 of each list are required. The SIQR calculates as

$$\text{SIQR} = \frac{Q_3 - Q_1}{2} \tag{7.12}$$

As index of central tendency, we use the median, which is the 50% quartile Q_2. For an eight-element list, Q_1 is found to be the 3rd element, Q_2 the 4th, and Q_3 the 6th element. Our calculation yields

	HOF	LOF-CLIQ
SIQR	3	1
Median	-4	-2

For HOF the mean delay varies by 3 percent points, whereas for LOF-CLIQ by just 1. The median is -4% for HOF, i.e. the results are averagely 4% worse than our Reference case. For LOF-CLIQ, it is just -2%.

Thus, considering short and long links, we conclude that LOF-CLIQ is superior to HOF for all cases that we simulated. However, the differences are small. We recall that HOF is an improper FC scheduling strategy for traffic with uniform destinations, unlike LOF-CLIQ. LOF-CLIQ is a strategy that performs well under bursty and non-bursty traffic with both uniform and non-uniform destination distributions.

Discussion

The strategies HOF and LOF have swapped their position in the performance graphs in comparison to uniform traffic. This is the most astonishing result. Considering the traffic pattern applied it can be explained. In a non-uniform traffic distribution one destination per FCD is constantly favored over the others. As there is also “normal” traffic on the same destination in other FCDs, the favored port will experience more contention. In a LOF strategy, a reception buffer with the highest occupancy would be considered last. Therefore in LOF, there are always credits available at the reception side for the favored port. As most of the traffic, however, is destined to this port, a large portion of the required bandwidth can not be provided due to credit under-run. This is in contrast to HOF. Here, this overloaded port is better supported, also yielding better results.

A remedy to the LOF for non-uniform traffic is LOF-CLIQ, where the status of the input queues is known at the reception side. This RXS scheduling strategy has now the possibility to improve its decision made on a pure LOF basis and return credits to those VOQs already awaiting them. Thus LOF-CLIQ is able to automatically detect changes in the spatial distribution of traffic, i.e. distribution of destination, and is able to react on it.

For such traffic patterns it is more important to return the credits to those VOQs that need them, than trying to improve performance with respect to the Reference by favoring destinations that have less traffic. With other words, in non-uniform traffic situations, the scheduling of the FC information that considers solely the state of the reception buffer is insufficient. In order to complete the picture, the state of the send buffers is required. In fact, for this traffic pattern, a strategy only based on CLIQ could be used delivering equally good results.

The strategies based on temporal or random properties perform poorly. This shows that for a proper return strategy the buffer states at both the sending at reception side are important.

Summary

We performed an analysis where we applied a non-uniform traffic pattern on the switch. The system parameters that were considered in this study are summarizes in Table 7.20. We additionally simulated the case $BS = 1$ and $BS = 10$. For the former we utilized a cross-point memory size of $M = 100\%$ RTT, while for the latter $M = 200\%$ RTT was chosen. We obtained performance results that show system delay over the unbalanced probability w, which describes the degree of non-uniformity. For instance $w = 0.0$ signifies uniform distributed destinations, whereas $w = 1.0$ signifies directed traffic.

Table 7.20: System parameters of the Evaluation of the Influence of Non-uniform Traffic

		System Size		
RTT	M[%RTT]	8×8	16×16	32×32
RTT=4	20			
	40			
	60			
	100	$\times$		$\times$
	140			
	200	$\times$		$\times$
RTT=64	20			
	40			
	60			
	100	$\times$		$\times$
	140			
	200	$\times$		$\times$

As a main result we saw the Reference to achieve the best results, closely followed by LOF-CLIQ and HOF. We performed an statistical analysis to find out that LOF-CLIQ performs better than HOF for the scenarios simulated. Furthermore, it was demonstrated that the results are empirically *independent of system size*.

We discussed the results and found that a RXS strategy based on random or temporal properties can be dismissed. Both are considered random with respect to the state of both the send buffers and reception buffers of a FCD. However, these states are important to make a good return decision. We also explained that in a non-uniform traffic pattern the state of the send buffers outweighs the importance of the reception buffer states. This is the reason why the LOF-CLIQ is able to outperform LOF. It is better than HOF, because it combines both, the benefits of LOF and is able to respect the credit requirements at the sending side.

Moreover, LOF-CLIQ is simpler than maintaining two algorithms (LOF and HOF) that have to be selected according to the switch environment or expected traffic pattern.

7.6.5 Impact of Traces of Multiprocessor Workloads

Motivation

The RXS strategies have been analyzed so far for traffic with uniformly and non-uniformly distributed destinations. These traffic patterns are synthetically generated. It is essential for our results to check them against the heterogeneous traffic patterns of real multi-processor workload as described in Subsection 2.5.2. The use of traces hereby was justified in Section 2.5.

Goal

We want to find out, how our five RXS strategies perform under the multi-processor workloads: FFT, MP3D, LU, Radix, and Water.

Experimental Set-up

We conduct our experiments for short links (RTT=4) as we have seen from the last sections that FC scheduling has more impact on systems with short links. We utilize a trace speed-up of 25 and 40 as defined in Subsection 2.5.2, to project future workloads.

These workloads suffer from the problem that their average load is really low. Therefore the differences in system delay are hardly visible. We therefore select the "severe credit under-flow" as our metric of interest. It determines the quality of the RXS strategies at a finer granularity. A severe credit under-flow situation occurs, if at any IA at least one packet is eligible, but is not dispatched due to missing feasibility information. This number thus indicates for how many packet cycles during the simulation in the presence of eligible packets, the upstream data-path scheduler was not able to schedule a packet and therefore left the up-stream link idle. A higher number yields a worse up-stream link utilization and potentially a worse performance. We put this number in relationship to the credit under-flow situations obtained from the Reference, similar to Eq. 7.10. Negative numbers refer therefore to relatively higher credit under-flow occurrences which is worse, while positive numbers represent lower such situations which is better.

We model a system size of $N = 4$, 16 and 32. As the workload is entirely deterministic no confidence on the results has to be obtained.

From the workloads that we modelled, we found that the Water application exhibits the same results for delay, as well as for credit under-flow situations. For better visibility of the results of the remaining workloads, we exempt Water from being displayed.

Results

In Figures 7.50 to 7.52 we see that for all parameters simulated the credit under-flow occurrence is higher for *any* RXS-strategy analyzed (negative percentages). This means that traffic patterns caused by multiprocessor workloads are less predictable than the patterns with a uniform or non-uniform destination distributions. For the latter a certain continuity or discontinuity of flows could always be assumed. This "knowledge" is no longer available. Nevertheless, HOF delivers the best overall results, coming closest to the Reference. LOF-CLIQ comes next. It shows equally good results for faster traces as shown in Figs. 7.50b) to 7.52b). For FFT is even surpasses the results of HOF. LOF-CLIQ is also superior on the 4-node system across all workloads (FFT, LU, and Radix) simulated. For slower traces, LOF-CLIQ is better than LOF, but roughly in the range of FIFO and RR, as can be seen in Figs. 7.50a) to 7.52a). An explanation for this behavior of LOF-CLIQ is a flaw in the simulation model that does not

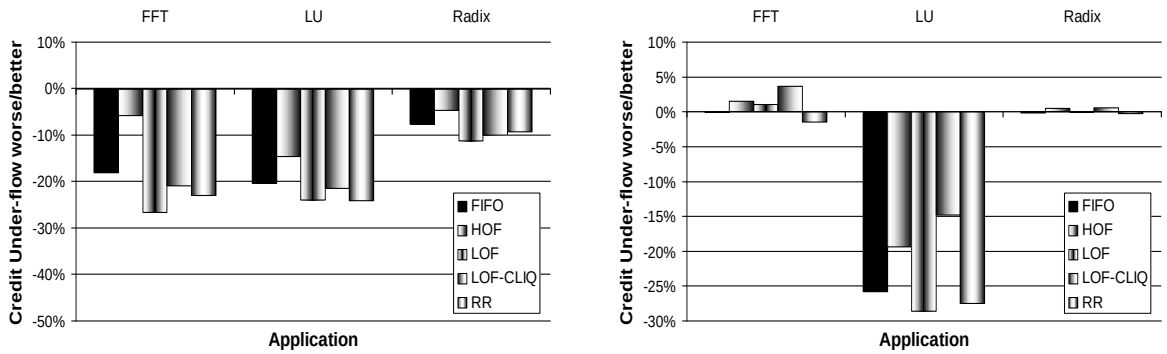


Figure 7.50: RXS Strategies Exposed to Traces, 4-Node Multiprocessor; (a) Trace-speed=25, (b) Trace-speed=40

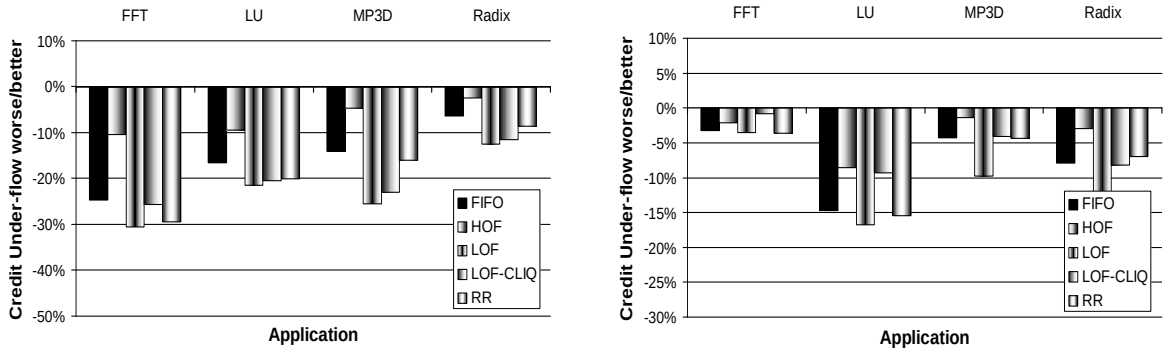


Figure 7.51: RXS Strategies Exposed to Traces, 16-Node Multiprocessor; (a) Trace-speed=25, (b) Trace-speed=40

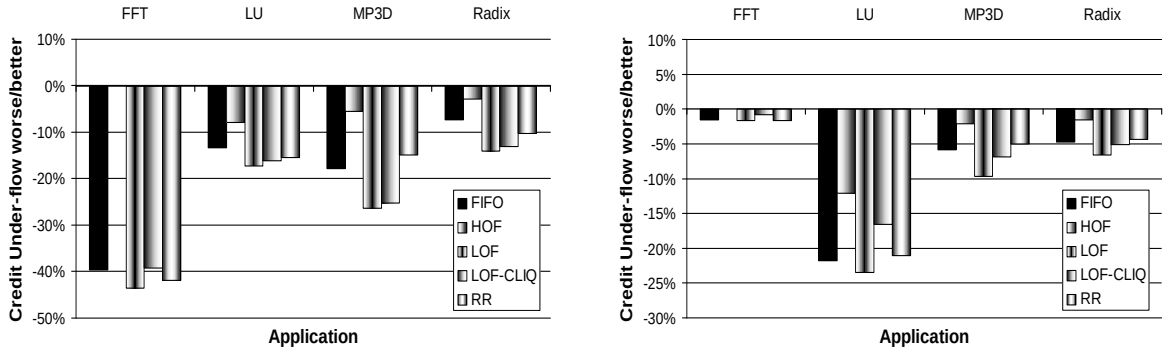


Figure 7.52: RXS Strategies Exposed to Traces, 32-Node Multiprocessor; (a) Trace-speed=25, (b) Trace-speed=40

transmit idle cells. These idle cells in turn were necessary to give the latest update of the input queue status under low loads. We believe that this is the reason why under more stress (trace speed of 40), LOF-CLIQ operates significantly better than under lower loads (trace speed of 25). For higher loads the status of the input queues is therefore more accurate.

Summary and Conclusion

This analysis shows that multiprocessor workloads, i.e traces thereof, are not as bandwidth demanding as for instance communication traffic. This is expressed by the almost equally good delay results independent of RXS-strategy. For this reason, we have chosen a metric that allows us to distinguish the schemes at a finer granularity. Credit under-flow is our means to determine the quality of the different RXS-schemes. Credit under-flow is a criteria to evaluate

the up-stream link utilization which directly impacts switch performance. We see that HOF and LOF-CLIQ again deliver the best results and come closest to the Reference case.

Chapter 8

Conclusions

The subject of this thesis has been the design of a scalable FC mechanism for buffered switches in interconnection networks such as communication and multiprocessor systems. FC is important for performance and scalability of interconnection networks. This FC scheme is the best so far due to its load-balancing properties that optimize performance for a set of paired send and receive buffers sharing a common link and low overhead. As an application example of this highly optimized flow-controlled link we have chosen a CICQ switch architecture with large round-trip time delays. The mechanism is employed at the ingress for SF internal FC between the adapters and the switch core. We have demonstrated its scalability and have proven by simulation its good performance and robustness by exposing the scheme to various different traffic patterns. We reviewed and classified existing FC schemes relevant for switch architectures. We quantified the trade-offs between FC complexity and FC overhead. We motivated the usage of a CICQ and relative-update credit FC as well as the utilization of RR data path schedulers.

This dissertation makes the following contributions to the design of packet-switches and interconnection networks:

- FC scheduling as a means to minimize FC overhead, resulting in a FC scheme that is universal to different switching environments and designed to endure future generations of switches.
- Definition of RXS scheduler strategies that implement FC scheduling and analysis of their impact on performance in terms of delay and throughput for a rich set of system parameters and traffic characteristics.
- Analysis of the switch dynamics in order to understand the relationship between QoS support, switching environment, and switch design, introducing the power as fairness metric.

8.1 FC scheduling

We have introduced the concept of FCDs and FCSDs to describe the structural relationship of switches and FC. It is an aggregation, i.e. a whole/part relationship, whereas the switch as a whole consists of multiple autonomous FCD parts. Each FCD itself consists of multiple FCSDs. This approach allowed us to reduce the complexity of the problem domain such that it suffices to optimize the behavior of one FCD. The optimization effort consists of minimizing FC overhead without a penalty in switch performance in terms of delay and throughput. We achieved this by means of a FC scheduling scheme which allows a reduction of the otherwise necessary

FC bandwidth of N to $\log N$ bits per FCD, where N is the switch dimension. We suggested to distribute the packet-scheduling activity between the sending and the receiving end of a FCD. A benefit of our approach is that the data-path scheduler at the sending side is relatively simple. Its objective is to provide fairness among eligible and feasible packets competing for link access. This can be achieved by a RR algorithm. The data-path scheduling decision is constrained from the receiving side by the RXS scheduler, which realizes FC scheduling [42]. In a CICQ switch architecture, the RXS scheduler is responsible for returning feasibility information of these cross-point queues for which it favors future packet arrivals. This is advantageous because during contention the switch can decide locally where future packet arrivals should be directed to. Therefore the RXS scheduler performs state information filtering in order not to overwhelm the data-path scheduler with injecting options and therefore increasing the probability of a packet selection that is detrimental to the overall switch performance. To avoid situations in which the switch requests packets from VOQs that have no packets, an additional information channel is provided between the sending and the packet-receiving side that transmits the status of the input queues, which we referred to as CLIQ. This additional channel increases the required FC bandwidth to $2 \log N$, which still scales much better than N. Overall, at low network loads, FC scheduling allows the switch to operate as an output-queued switch, because traffic may proceed forward without delay. During periods of high network loads, it allows the switch to operate as an input-queued switch, because traffic is pulled from the reception side. The scheme is scalable, universal and applies well to different switching environments because of its flexibility in reacting to highly loaded – but sparsely scattered – as well as densely scattered flows. FC scheduling and the CLIQ concept are secured by a patent application [43].

8.2 RXS Scheduler Strategies

We have proposed various FC scheduling strategies. The strategies analyzed are based on random, temporal, and spatial properties. We found that strategies based on temporal and random properties do not perform well owing to their independence of the switch's buffer state. Spatial strategies, on the other hand, have shown a performance impact.

We derived the following conclusions:

- Under uniform traffic, the LOF and LOF-CLIQ strategies achieved the best results. They surpassed the results of the Reference, i.e. the unscheduled FC channel of capacity N, by up to 12% for memory sizes of 60 and 100% RTT. LOF achieved a median of 8.4% performance gain for these memory sizes and slightly, normal, and heavily bursty traffic, while LOF-CLIQ achieved 6.8%. The LOF strategy favors the return of those credits whose corresponding reception buffer is close to data under-run. This achieves a lower average receiver-buffer utilization, hence a reduced switch delay and overall system delay. This strategy results in much better contention resolution. It is superior to the LOF-CLIQ strategy, because the availability of packets can be tacitly assumed owing to the uniformity of destinations. For LOF-CLIQ a fine tuning of the CLIQ parameters was required to achieve equal results. HOF delivers even worse performance than the Reference as well as the temporal and random strategies. The LOF-CLIQ strategy works as LOF, but in addition takes the occupancy information of the VOQ at the input into account.

- For non-uniform traffic the HOF and LOF-CLIQ strategies give the best results. HOF is able to exploit the non-uniform nature of destinations successfully as it favors the return of those credits whose corresponding reception buffers are close to credit under-run. Under uniform traffic, this would stimulate more contention to an already contended cross-point.

Under such circumstances this behavior is beneficial, as unfavored destinations are rarely utilized. As a consequence, HOF favors highly loaded, but sparse flows. Under these traffic assumption, we see the real benefit of CLIQ. Whereas LOF is inferior to HOF here, HOF is inferior to LOF for uniform traffic. LOF-CLIQ outperforms HOF for non-uniform traffic and is as good as LOF for uniform traffic. This demonstrates the usefulness of our proposed protocol. In general the Reference case delivers the best results, followed by LOF-CLIQ, which itself is closely followed by HOF.

- Using traces of multiprocessor workloads we did not see much performance difference irrespective of the RXS scheduling strategies applied, because of the low loads the workloads generate on average. For evaluation purposes, we used a more precise criterion, namely that of severe credit under-run. With this metric we saw significant differences that also largely depend on the workload. Here, HOF and LOF-CLIQ deliver the best results, with HOF leading.

- Impact of system parameters. We found the results of FC scheduling to be independent of system size. We saw that shorter links exhibit larger performance differences. This means that a strategy that already has performance advantages in the case of long links will provide even more performance benefits for a system with short links. This same amplifying effect was observed for strategies that perform poorly. They yielded even worse results.

  Under uniform traffic assumptions the cross-point memory size can be reduced to about 60% RTT. Although this signifies that a single flow of an arbitrary input and output pair could only send at 60% of the link capacity in the absence of output contention, it saves 40% buffer space. This buffer space could be used to QoS support. With LOF and LOF-CLIQ, the performance results are as good as those of the Reference for a memory size of 200% RTT. The benefits are buffer savings that could be used, for instance, to support more traffic classes.

In summary, with LOF-CLIQ we have achieved a FC scheduling algorithm that consumes low FC bandwidth of $2 \log N$ per FCD and allows the SF to perform equally well or even better than an unscheduled control path with bandwidth of N. This algorithm eliminates FC bandwidth becoming a bottleneck when scaling to faster and larger SFs. It further is a scheme that is endurable, robust, and applicable in both communication and multiprocessor environments as it shows good performance results for a variety of traffic patterns, such as uniform, non-uniform and traces of MP workloads.

8.3 Switch Dynamics and Fairness

We have provided insight into the switch behavior at the output side, and analyzed two very basic output-queue scheduling algorithms, RR and OCF. By excluding queue reordering, these two algorithms map directly onto a CIOQ and CICQ architectures. We performed two analyses:

- **Static input loads.** We showed that for OCF scheduling or, equivalently, for a CIOQ switch architecture, a high correlation exists between highly and lightly loaded flows originating from different inputs but routed to the same output. This correlation is much lower in the case of RR scheduling, i.e. a CICQ architecture. We demonstrated that aggregate mean delay values are inappropriate to render differences in the serving quality visible. We therefore proposed the power as a metric for this purpose, which is expressive and easy to calculate.

- **Dynamic input loads.** We showed that the reaction time to changing loads – the dynamics of the switch – is much larger for OCF than it is for a RR scheme.

We concluded that in a non-cooperative environment a CICQ is superior to a CIOQ switch architecture in terms of input fairness. A CICQ with a RR output, i.e. cross-point queue scheduler, or one of its variants such as WRR, DRR, is thus able to support the growing QoS demands of switches much better. Such an architecture allows single flows to be controlled.

However, for cooperative environments, a CIOQ seems to be more appropriate. This means that a possible oversubscription of an input-output pair is not something the switch should take countermeasures against, as it is interpreted as an intention necessary for the overall good of the entity creating traffic. An example of such an entity is, for instance, a multiprocessor workload.

8.4 Future Work

The FC scheduling strategies proposed in this thesis are based on first-order effects, such as queue fill level. More complete schemes would also take second-order – derivatives – aspects into account, such as filling or drain rates. Future work should also extend to the implementational issues of how FC scheduling has to cooperate with output-queue scheduling, in order to guarantee delay bounds and link bandwidth. The influence of output-queue scheduling algorithms in multiprocessor environments should be applied to an execution-driven simulation environment, taking the interdependency of computation and communication into account.

Appendix A

Definitions

As introduced throughout the text of this chapter and this thesis, we will use the following terms:

Packet A Packet is a datagram unit of configurable size, consisting of a header and a payload.

Routing Routing signifies the function of path determination according to parameters that are mostly network protocol dependent.

Switching Switching is the function of executing a previously made routing decision

Ingress Ingress is this side of the switch, which is coming from the SA (from adapter to switch)

Egress Egress is this side of the switch, which is going to the SA (from switch to adapter)

Contention Contention naturally happens within a switch. It occurs, if two or more inputs send traffic to the same output at the same time. Contention not resolved within due time might be the cause of downstream congestion

Congestion Congestion ripples through all downstream queueing points and leads to the filling of buffers. It occurs, if the *upstream* node is not able to accept the current traffic arrival rate and signals to either stop or throttle it.

Workload Workload is the amount of work, which is given by an application, a closed system, i.e. multiprocessor system, has to cope with. From our view point, a workload is the cause of a specific traffic pattern, the communication subsystem has to deal with.

Downstream Downstream is the side closer to the final destination of a packet along its data path.

Upstream Upstream is the side closer to the sending source of a packet along its data path.

Data Path The data path or data channel is the path of a user data packet through the network from the source to the destination, i.e. from upstream to downstream nodes.

Control Path The control path or control channel is the path of a control information travelling in the opposite direction of the data path. It is often also referred to as *reverse path*.

Forward Path See Data Path.

Reverse Path See Control Path.

Node A node enqueues, modifies, and forwards packets according to some given rules. It is therefore an entity that consists of at least one queue and some logic that determines both the service discipline and the modification algorithm.

Queueing Point See Node.

Blocking Blocking occurs, if the feasibility information is removed.

Unblocking Unblocking is the fact of receiving the feasibility information.

Flow A flow is a stream of packets of a given service class between arbitrary input and output pairs. Sequence order is maintained and the same grade of service is provided while traversing the SF.

Appendix B

Why Cables? - An Example, IBM PowerPRS

Let's look at an example, IBM's current switching family, the IBM PowerPRS [40], to demonstrate a fundamental requirement of future switching. This requirement is determined by physical constrains: 1) limited number of signal I/Os per chip 2) limited number of boards per backplane.

In order to support a 32×32 switch with a data rate of 2.0 Gb/s, there are 128 signal I/Os necessary using for instance IBM's Unilink technology that runs at 2.5 Gb/s, to account for the 25% overhead imposed by the 8/10 bit coding. By doing the calculation, a switch of an aggregate throughput of 64Gb/s is feasible on a single chip today. When scaling to a 32×32 switch supporting OC-192c (10 Gb/s) port speed, each switch port needs to be able to do 16 Gb/s accounting for all the necessary speed-up, i.e. overhead by segmentation and reassembly and the switch packet header, or 20 Gb/s at the physical level. This is equivalent to 8 Unilink interfaces per port. In total this configuration requires 1024 signal I/Os just for getting the data on and off the switch. IBM needs 8 chips to do this, which are mounted on two switch cards. The adapters still can be connected via the backplane. Hence, a single switch card can do 256 Gb/s. Taking into account that for high availability, i.e. reaching 99.999% availability over one year (about 5 minutes of down-time per year), a redundant system is included, the number of boards is increased in this case to four. There are 20 slots on a backplane, four serving for the switch cards, and 16 for the adapters (2 switch adapters per card).

Let's see the physical constraint when scaling to higher speeds. In order to support OC-768 (40 Gb/s) ports, there would be 20 Unilink interfaces necessary per port. Again for a 32×32 switch, this would require 2560 signal I/Os, and would be roughly a 2 Tb/s switching system. As we have seen before, switching cards of 256 Gb/s are feasible. As a consequence, we needed 8 such switching cards, or 16 for the redundant switching system. As a consequence, there is no more room for the adapter cards to also connect to the same backplane. By using cost efficient standard board and rack sizes, connectivity is not feasible any more over a backplane. This problem is also not solvable with technological advances, for instances if 5Gb/s links would be available.

An attractive engineering solution is to use cables instead, as shown in Fig. 2.4.

Appendix C

Implication of dropping packets

We have three basic arguments against dropping of packets at the SF as an alternative to FC. The first argument is that dropping of packets increases overall network load, because dropped packets and possibly frames need to be resent. This is in general unwanted. Therefore, mechanisms at the NP level are in place that try to stop packets from being dropped as early as possible. The argumentation is therefore, if this is tried to be avoided at the NP level, which costs FC and overhead, then why not do it at the SF level and offer this property as a service to the NP?

The second argument assumes that packets are only dropped in cases of severe congestion. It would therefore be a means for the SF to release resources to accept further packets. This argumentation assumes that the NP would pass on possible drop information and therefore the switch was only allowed to drop packets that were marked in such a way (“dropable”). If there was a stream of dropable and non-dropable packets then the SF could not rely on the fact that there were always enough dropable packets available to free up congested resources, if there was a need to. The only way therefore, that assures non-dropable packets not to be dropped is a proper FC mechanism in place.

Therefore a lossless SF by means of FC is more resource efficient and also a guaranteed way of supporting lossless traffic classes.

Dropping of packets of lossy traffic classes at the SF level and maintaining a FC mechanism for lossless traffic at the same time involves a lot of additional overhead. This overhead is not easily justified, because overhead is costly and the NP has much more protocol information to better judge, which packets to drop or not.

The third argument is that dropping packets is not acceptable for computer interconnects, because a dropped packet with system relevant data may easily hang-up the system. Therefore, it is important for a unified architecture to have the lossless property as a prerequisite.

Appendix D

Bursty Traffic Model

The burst size BS is mathematically spoken the average time spent in ON state according to Fig. 2.9. In the following, we will show how to realize a bursty source using a Markov-chain model.

$$BS = \overline{S_{\mathrm{ON}}} \tag{D.1}$$

The probability of being in state ON for n consecutive cycles $P(S_{\mathrm{ON}} = n)$ is geometrically distributed and is determined by the probability of success p_{11} of that state:

$$P(S_{\mathrm{ON}} = n) = p_{10}{p_{11}}^{n-1} = p_{10}(1-p_{10})^{n-1} \tag{D.2}$$

Once the ON state has been reached, at least one cell will be generated, i.e. $n \geq 1$. Hence, the average duration of the ON state

$$\begin{aligned}\overline{S_{\mathrm{ON}}} = \sum_{n=1}^{\infty} nP(S_{\mathrm{ON}} = n) = \sum_{n=1}^{\infty} np_{10}(1-p_{10})^{n-1} = p_{10}\sum_{n=1}^{\infty} n(1-p_{10})^{n-1} = \\ \frac{p_{10}}{1-p_{10}}\sum_{n=1}^{\infty} n(1-p_{10})^{n} = \frac{p_{10}}{1-p_{10}}\frac{1-p_{10}}{(1-(1-p_{10}))^2} = \frac{p_{10}}{(p_{10})^2} = \frac{1}{p_{10}}.\end{aligned} \tag{D.3}$$

For this reason the burst length BS is given by Eqs. D.1 and D.3

$$BS = \frac{1}{p_{10}} \tag{D.4}$$

Once the OFF state has been reached, it may be necessary to immediately leave this state and go back to the ON state with a new destination selected for the next burst, i.e. $n \geq 0$. The case of immediate return is necessary at a load of $L = 100\%$.

The average duration of the OFF state is therefore given by

$$\begin{aligned}\overline{S_{\mathrm{OFF}}} = \sum_{n=0}^{\infty} nP(S_{\mathrm{OFF}} = n) = \sum_{n=0}^{\infty} np_{01}(1-p_{01})^{n} = \\ p_{01}\sum_{n=0}^{\infty} n(1-p_{01})^{n} = p_{01}\frac{1-p_{01}}{(1-(1-p_{01}))^2} = \frac{1-p_{01}}{p_{01}}.\end{aligned} \tag{D.5}$$

Due to the line rates $\lambda_{\mathrm{ON}} = 1$ and $\lambda_{\mathrm{OFF}} = 0$, the average load L can be calculated by the ratio of the average duration of the ON to the average duration of ON and OFF states, which itself are given by Eqs. D.1 and D.5:

$$L = \frac{\overline{S_{\mathrm{ON}}}}{\overline{S_{\mathrm{ON}}} + \overline{S_{\mathrm{OFF}}}} = \frac{BS}{BS + \frac{1}{p_{01}} + 1} \tag{D.6}$$

With the load L and the average burst size BS given, the state transition probability p_{01} is found to be

$$p_{01} = \frac{1}{BS(\frac{1}{L} - 1) + 1} \tag{D.7}$$

The transition probability p_{10} has already been determined in Eq. D.4 as

$$p_{10} = \frac{1}{BS} \tag{D.8}$$

Appendix E

Description of the Multiprocessor Applications

The five applications of which traces were captured to model multiprocessor workload and their expected communication characteristics are here described. A detailed description of each application or application kernel is found in [71], [70] and the references therein.

FFT

FFT is the fast Fourier Transformation, which transforms n complex data points. It typically shows a high communication rate, uses all-to-all communication and exhibits a regular communication pattern. It consists of three phases during execution. The first and the last perform the radix-2 butterfly, whereas the middle phase is the communication phase. It has a high spatial locality, which means that large caches reduce the communication rate dramatically during the two radix-2 butterfly-phases.

For our chosen cache geometry, it has a communication to computation ratio of about 1 byte/FLOP [71], which is quite high. Nevertheless, an average communication to computation ratio does not reveal much about the timely behavior of communication, of which we show a sample in Fig. E.1 from our RSIM traces. This figure shows the overall communication volume, counted in number of messages that the network needs to transmit per processor cycle. Here, neither the type of message, nor the message size are distinguished. The graph shows a time window of the trace between processor cycles 78'000'000 and 86'000'000, which has the most activity. All generated messages of all processors are counted per $s_p = 10'000$ processor cycles. It is clearly seen that the communication volume varies a lot. There are periods of inactivity and periods of high communication, with up to 3500 messages. With message sizes 16 bytes and an expected average message length of 32 bytes, this translates into a peak bandwidth requirement of 3.4 GB/s for the given $f = 300$ MHz processors. The bucket that counts all messages during the 10'000 processor cycles is $t_{bucket} = \frac{s_p}{f} = 0.033$ ms long.

Water

The water application evaluates forces and potentials that occur over time in a system of water molecules. It typically has a low communication rate and uses nearest neighbor communication. The data objects being moved between processors are records that represent molecules. These objects are large and a large portion of the cache-lines being moved are actually used. Therefore, not much communication is necessary, which is also found in the measured communication volume over time in Fig. E.2. There is a peak of 1200 messages per 100'000 processor cycles in the beginning. The trace then continues with an average of 25 messages.

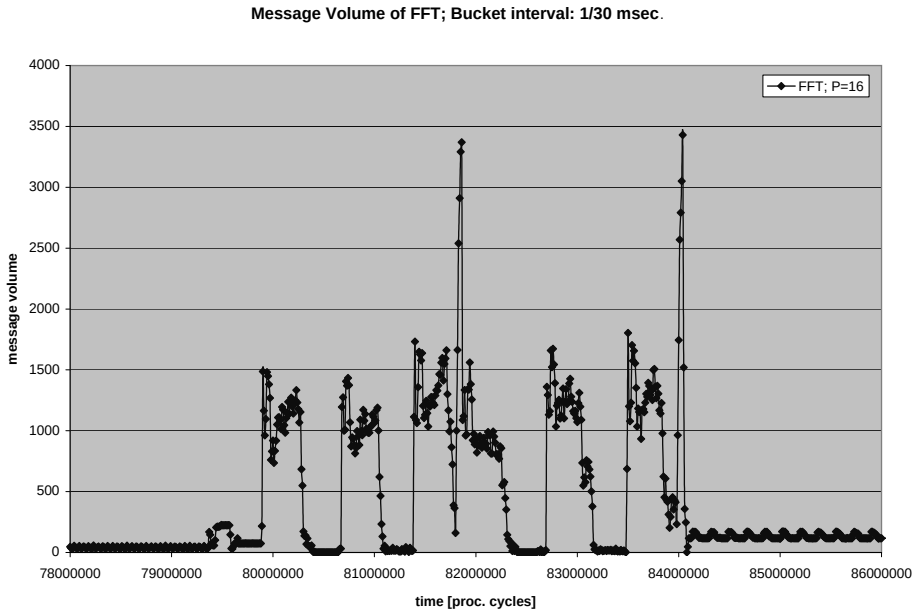


Figure E.1: FFT Trace File

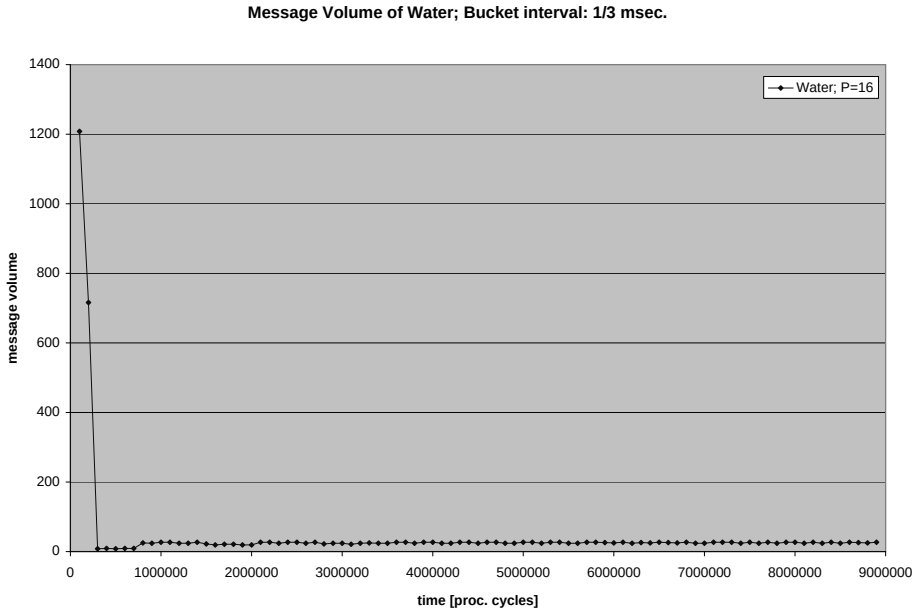


Figure E.2: Water Trace File

Radix

Radix is an iterative sorting algorithm with a high communication rate and all-to-all communication. Synchronization is performed by writes. For this reason radix shows more writes then reads, which is contrary to all other applications. The communication volume profile shows periods of constant high communication, but also communication holes and in the end a low communication rate. The trace activity is shown in Fig. E.3 which shows sustained bursts of high activity for about half the trace time. The remaining part shows a sustained low activity.

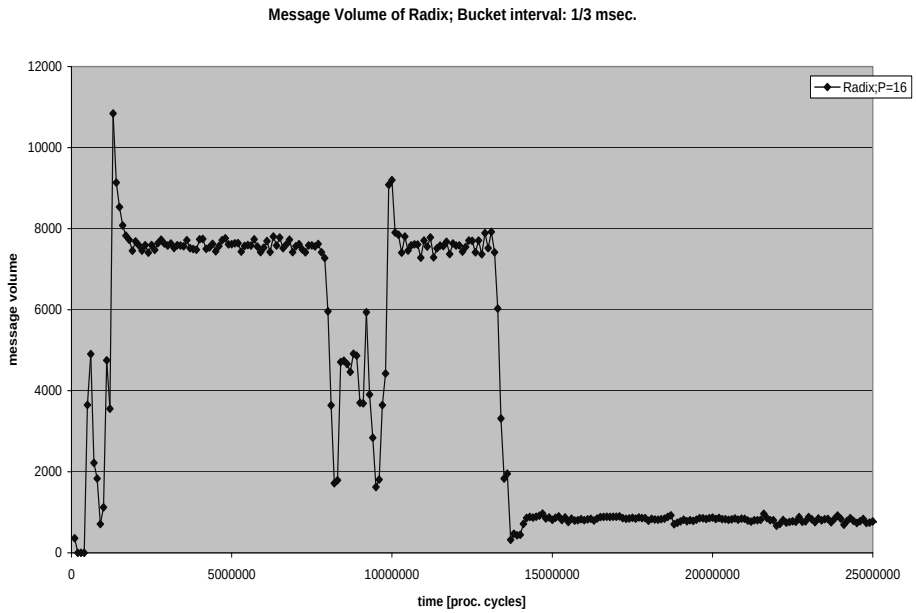


Figure E.3: Radix Trace File

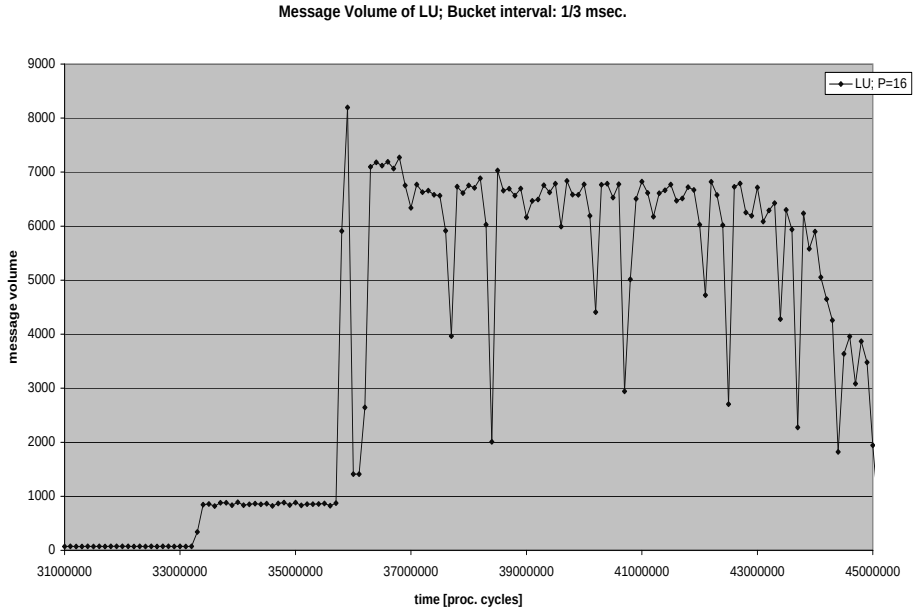


Figure E.4: LU Trace File

LU

The LU kernel factors a dense matrix into the product of a lower (L.) triangular and an upper (.U) triangular matrix, which also explains the LU abbreviation. The communication volume is distributed over time according to Fig. E.4. This graph also shows periods of low communication and periods of rather high and bursty communication.

MP3D

MP3D simulates a three-dimensional wind tunnel using particle-based techniques. It is used to study the shock waves created as an object flies at high speed through the upper atmosphere. It uses data objects smaller than a 64 byte cache-line. Similar as FFT and Radix, MP3D has

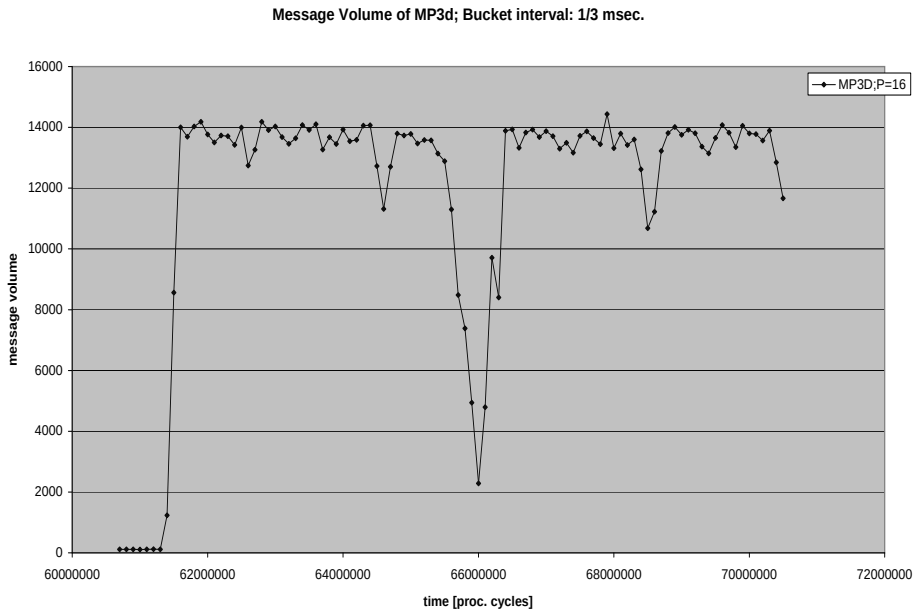


Figure E.5: MP3D Trace File

a high communication rate. It has two high communication phases that are interrupted in the beginning and in the middle.

Bibliography

[1] M. Katevenis, D. Serpanos, and P. Vatsolaki, "ATLAS I: A General-Purpose, Single-Chip ATM Switch with Credit-Based Flow Control," in *Proceedings of IEEE IV. Hot Interconnects Symposium*, (Stanford, USA), August 1996.

[2] G. Kornaros, D. Pnevmatiatos, P. Vatsolaki, G. Kalokerinos, C. Xanthaki, D. Mavroidis, D. Serpanos, and M. Katevenis, "Implementation of ATLAS I: A Single-Chip ATM Switch with Backpressure," in *Proceedings of IEEE VI. Hot Interconnects Symposium*, (Stanford, USA), pp. 85–96, August 1998.

[3] M. Katevenis and P. Vatsolaki, "ATLAS I: A Single-Chip ATM Switch with HIC Links and Multi-Lane Back-Pressure," in *EMSYS 96 Conference*, (Berlin, Germany), September 1996.

[4] R. Schoenen, G. Post, and A. Müller, "Analysis and Dimensioning of Credit-Based Flow Control for the ABR Service in ATM Networks," in *Proceedings of IEEE INFOCOM*, vol. 4, pp. 2399–2404, 1998.

[5] R. Schoenen and A. Dahlhoff, "Closed Loop Credit-Based Flow Control with Internal Backpressure in Input and Output Queued Switches," in *Proc. of IEEE High Performance Switching and Routing*, pp. 195–203, 2000.

[6] A.C. Kam and K.-Y. Siu, "Linear-Complexity Algorithms for QoS Support in Input-Queued Switches with No Speedup," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1040–1056, June 1999.

[7] M. Katevenis, "Buffer Requirements of Credit-Based Flow Control when a Minimum Draining Rate is Guaranteed," in *HPCS'97 (4th IEEE Workshop Arch. Impl. High Perf. Communications Subsystems)*, (Chalkidiki, Greece), June 1997.

[8] H.T. Kung and A. Chapmann, "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks: A Summary," in *Proceedings of the International Conference of Network Protocols*, pp. 116–127, October 1993.

[9] H.T. Kung, . Morris, T. Charuhas, and D. Lin, "Use of link-by-link flow control in maximizing atm networks performance: Simulation results," in *Proceedings of IEEE Hot Interconnects Symposium*, (Palo Alto, USA), August 1993.

[10] C. Özveren, R. Simcoe, and G. Varghese, "Reliable and efficient hop-by-hop flow control," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 642–650, May 1995.

[11] F. Silla and J. Duato, "On the Use of Virtual Channels in Networks of Workstations with Irregular Topology," in *Proceedings of the 1997 Parallel Computing, Routing and Communication Workshop (PCRCW'97)*, June 1997.

[12] J. Turner, "Gigabit Kits Course Switch Architecture." http://www.arl.wustl.edu/ jst/gigatech/kits.html, April 1998.

[13] P. Mishra and H. Kanakia, "A Hop by Hop Rate-based Congestion Control Scheme," in *Proceedings of ACM SIGCOMM'92*, pp. 112–123, October 1992.

[14] F. Bonomi, D. Mitra, and J.B. Seery, "Adaptive Algorithms for Feedback-Based Flow Control in High Speed, Wide-Area ATM Networks," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 7, pp. 1267–1283, 1995.

[15] X. Yang, "A Model for Window Based Flow Control in Packet-Switched Networks," in *Proceedings of IEEE INFOCOM*, vol. 2, pp. 423–430, 1999.

[16] J.D. Atkins, "Path Control: The Transport Network of SNA," *IEEE Transactions on Communications*, vol. COM-28, pp. 527–538, April 1980.

[17] T.V. Lakshman, U. Madhow, and B. Suter, "Window-based Error Recovery and Flow Control with a Slow Acknowledgment Channel: A Study of TCP/IP Performance," in *Proceedings*, pp. 1199–1209, 1997.

[18] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, pp. 1465–1480, October 1995.

[19] H. Ohsaki, M. Murata, T. Ushio, and j. . I. m. . A. y. . . p. . . H. Miyahara, title = A Control Theoretical Analysis of a Window-based Flow Control Mechanism in TCP/IP Networks

[20] H. Ohsaki, M. Murata, T. Ushio, and H. Miyahara, "A Control Theoretical Approach to a Window-based Flow Control Mechanism with Explicit Congestion Notification," in *38th IEEE Conference on Decision and Control*, December 1999.

[21] E. Bowen, C. Jeffries, L. Kencl, A. Kind, and R. Pletka, "Bandwidth Allocation for Non-Responsive Flows with Active Queue Management," in *Proceedings of the International Zurich Seminar on Broadband Communications*, pp. 13.1 – 13.6, February 2002.

[22] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, August 1993.

[23] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Transactions on Communications*, vol. COM-28, pp. 553–574, April 1980.

[24] "Quantum Flow Control." FCC-SPEC-95-1, July 1995.

[25] "CSIX-L1: Common Switch Interface Specification-L1." CSIX, 2130 Hanover St., Palo Alto, CA USA, May 2000.

[26] "Virtual Interface Architecture Specification, Version 1.0." Compaq Computer, Intel, and Microsoft Corporation, December 1997.

[27] Y.A. Korilis and A.A. Lazar, "Why is flow control hard: Optimality, fairness, partial and delayed information," in *2nd ORSA Conference on Telecommunications*, (Boca Raton, FL), March 1992.

[28] I. Iliadis, "A New Feedback Congestion Control Policy for Long Propagation Delays," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1284–1295, September 1995.

[29] W. Bux, K. Kümmerle, and H.L. Truong, "Balanced HDLC Procedures: A Performance Analysis," *IEEE Transactions on Communications*, vol. COM-28, pp. 1889–1980, November 1980.

[30] K. Kant, "Flow Control Mechanisms for SAAL Links," in *Proceedings of IEEE International Conference on Broadband Communications*, pp. 173–184, 1996.

[31] K. Kant, "Analysis of Delay Performance of ATM Signaling Link," in *Proceedings of IEEE INFOCOM*, vol. 3, pp. 1146–1153, 1995.

[32] L.-S. Peh and W.J. Dally, "Flit-Reservation Flow Control," in *Proceedings of the sixth International Conference on High Performance Computer Architecture*, pp. 73–84, 1999.

[33] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, pp. 194–205, March 1992.

[34] W. Denzel and I. Iliadis, "Performance of Packet Switches with Input and Output Queueing," in *Proceedings of ICC/SUPERCOMM'90*, (Atlanta, GA.), pp. 316.3.1–316.3.7, 1990.

[35] I. Iliadis, "Performance of a Packet Switch with Shared Buffer and Input Queueing," in *Proceedings of 13. International Teletraffic Congress*, (Copenhagen, Denmark), pp. 911–916, 1991.

[36] W.E. Denzel, A.P.J. Engbersen, I. Iliadis, and G. Karlsson, "A Highly Modular Packet Switch For GB/S Rates," in *Proceedings of XIV. International Switching Symposium*, (Yokohama, Japan), October 1992.

[37] C. Minkenberg and T. Engbersen, "A Combined Input- and Output-Queued Packet-Switch System Based on PRIZMA Switch-on-a-Chip Technology," *IEEE Communications Magazine*, vol. 38, pp. 70–77, December 2000.

[38] C. Minkenberg, T. Engbersen, and M. Colmant, "A Robust Switch Architecture for Bursty Traffic," in *Proceedings of the International Zurich Seminar on Broadband Communications*, (Zurich, Switzerland), February 2000.

[39] H. Alaiwan, "IBM 8265 ATM Backbone Switch Hardware Architecture," *Computer Networks*, vol. 31, pp. 527–540, March 1999.

[40] http://www-3.ibm.com/chips/products/wired.

[41] F. Abel, C. Minkenberg, R.P. Luijten, M. Gusat, and I. Iliadis, "A Four-Terabit Single-Stage Packet Switch with Large Round-Trip Time Support," in *Proceedings Hot Interconnects'10, Symposium on High Performance Interconnects (HOT-I'02)*, (Palo Alto, CA.), August 2002.

[42] F. Gramsamer, M. Gusat, and R.P. Luijten, "Optimizing Flow Control for Buffered Switches," in *Proceedings International Conference on Computer Communications Networks (ICCCN'02)*, (Miami, Fl.), October 2002.

[43] F. Gramsamer, J. Duato, T. Engbersen, M. Gusat, and M.Verhappen, “Improvements in Flow-Control Scheduling: Scalable Closed-Loop Flow-Control Based on Near-Optimal Revers-Path Scheduling and Scheduling Hints.” Patent Application, IBM Docket Nr.: CH9-20020004EP1, November 2001.

[44] M. Colmant, F. Gramsamer, and C. Minkenberg, “Switching Arrangement and Method.” Patent Application, IBM Docket Nr.: CH9-19990011EP1, May 1999.

[45] M. Gusat, F. Abel, F. Gramsamer, R. Luijten, C. Minkenberg, and M. Verhappen, “Stability of CIOQ Switches with Finite Buffers and Non-Neglibible Round-Trip Time,” in *Proceedings International Conference on Computer Communications Networks (ICCCN’02)*, (Miami, Fl.), October 2002.

[46] T. Javidi, R. Magill, and T. Hrabik, “A High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric,” in *ICC*, (Helsinki,Finland), June, 14-16 2001.

[47] M. Nabeshima, “Performance Evaluation of a Combined Input- and Crosspoint-Queued Switch,” *IEICE TRANS. COMMUN.*, vol. E83-B, pp. 737–741, March 2000.

[48] D. C. Stephens and H. Zhang, “Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture,” in *17th Annual Joint Conference of the IEEE Computer and Communication Society Proceedings*, vol. 1, pp. 282–290, 1998.

[49] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Jonathan Chao, “CIXB-1: Combined Input-One-cell-Crosspoint Buffered Switch,” in *Proc. 2001 IEEE Workshop on High Performance Switching and Routing*, (Dallas, TX), pp. 324–329, May 2001.

[50] R. Rojas-Cessa, E. Oki, and H. Jonathan Chao, “CIXOB-k: Combined Input-Crosspoint-Output Buffered Packet Switch,” in *Proc. GLOBECOM’01*, vol. 4, pp. 2654–2660.

[51] S. Keshav and R. Sharma, “Issues and Trends in Router Design,” *IEEE Communication Magazine*, pp. 144–151, May 1998.

[52] V.P. Kumar, T.V. Lakshman, and D. Stiliadis, “Beyond Best Effort: Router Architectures for Differentiated Services of Tomorrow’s Internet,” *IEEE Communication Magazine*, pp. 152–164, May 1998.

[53] L. Bhuyan, N. Ni, and M. Pirvu, “The Impact of Link Arbitration on Switch Performance,” in *Proceedings of the 5th International Symposium on High Performance Computer Architecture (HPCA)*, 1999.

[54] A. Demers, S. Keshav, and S. Shenker, “Analysis and Simulation of a Fair Queueing Algorithm,” in *Proceedings of ACM SIGCOMM’89*, pp. 1–12, September 1989.

[55] J. Nagle, “On Packet Switches with Infinite Storage,” *IEEE Transactions on Communications*, vol. 35, pp. 435–438, 1987.

[56] P. McKenney, “Stochastic Fairness Queueing,” *Internetworking: Research and Experience*, vol. 2, pp. 113–131, 1991.

[57] M. Shreedhar and G. Varghese, “Efficient Fair Queueing using Deficit Round Robin,” in *Proceedings of ACM SIGCOMM’95*, pp. 1–22, October 1995.

[58] E.L. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1024–1039, September 1991.

[59] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, June 1993.

[60] F.M. Chiussi and A. Francini, "Providing QoS Guarantees in Packet Switches," in *Proceedings IEEE Globecomm'99*, (Rio de Janeiro, Brazil), pp. 1582–1590, December 1999.

[61] F.M. Chiussi, A. Francini, D.A. Khotimsky, and S. Krishnan, "Feedback Control in a Distributed Scheduling Architecture," in *Proceedings IEEE Globecomm'2000*, (San Francisco, Ca.), October 2000.

[62] F.M. Chiussi and A. Francini, "A Distributed Scheduling Architecture for Scalable Packet Switches," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2665–2683, December 2000.

[63] F.M. Chiussi, J.G. Kneuer, and V.P. Kumar, "Low-Cost Scalable Switching Solutions for Broadband Networking: The ATLANTA Chipset," *IEEE Communications Magazine*, vol. 35, pp. 44–53, December 1997.

[64] T. Lovett and R. Clapp, "STiNG: A CC-NUMA Computer System for the Commercial Marketplace," in *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, May 1996.

[65] D. Kotz and N. Nieuwejaar, "Dynamic File-Access Characteristics of a Production Parallel Scientific Workload," in *IEEE Supercomputing*, pp. 640–649, 1994.

[66] St. Emerson, "Evaluation of a Data Communication Model for Switched Fibre Channel," *IEEE Network*, pp. 38–44, November/December 1995.

[67] K. Hwang, Ch .Wand, and C.-L. Wang, "Evaluation MPI Collective Communication on the SP-2, T3D, and Paragon Multicomputer," in *Proceedings of 3rd International Symposium on High Performance Computer Architecture, (HPCA-3)*, pp. 106 – 116, February 1997.

[68] E. Boyd, J.D. Wellmann, S. Abraham, and E. Davidson, "Evaluating the Communication Performance of MPPs Using Synthetic Sparse Matrix Multiplication Workloads," in *Proceedings of the 7th ACM International Conference on Supercomputers*, (Tokyo, Japan), July 1993.

[69] D. Jiang, X. Yu, S. Kumar, A. Bilas, and J.P. Singh, "Application Scaling under Shared Virtual Memory on a Cluster of SMPs." `f`rom University of Toronto. Find out where it was published. tbr.

[70] J.P. Singh, W.-D. Weber, and A. Gupta, "SPLASH: Stanford Parallel Applications for Shared-Memory," *Computer Architecture News*, vol. 20, pp. 5–44, March 1992.

[71] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *22nd Annual International Symposium on Computer Architecture*, pp. 24–36, 1995.

[72] F. Gramsamer, "Bimodal Traffic for Computer Interconnects," in *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'01)*, (Orlando, USA), pp. 583–589, July 2001.

[73] V.S. Pai, P. Ranganthan, and S.V. Adve, "RSIM Reference Manual, Version 1.0," Tech. Rep. 9705, Dept. of Electrical And Computer Engineering, Rice University, 6100 South Main, Houston, Texas 77005, August 1997.

[74] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee, "On Credibility of Simulation Studies of Telecommunication Networks," *IEEE Communication Magazine*, pp. 132–139, January 2002.

[75] D.E. Culler, R.M. Karp, D.A. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T.von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," in *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 262–273, 1993.

[76] R.P. Martin, A.M. Vahdat, D.E. Culler, and T.E. Anderson, "Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture," in *International Symposium on Computer Architecture (ISCA)*, (Denver, CO., USA), pp. 85–97, 1997.

[77] A. Sivasubramaniam, "Towards a Communication Characterization Methodology for Parallel Applications," pp. 310–319, 1997.

[78] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," vol. 2, pp. 1–15, February 1994.

[79] J.G. Dai and B.Prabhakar, "The Throughput of Data Switches with and without Speedup," in *Proceedings INFOCOM 2000*, vol. 2, (Tel Aviv, Israel), pp. 556–564, March 2000.

[80] K.J. Christensen, "Design and Evaluation of a Parallel-Polled Virtual Output Queued Switch," in *ICC*, (Helsinki,Finland), June 2001.

[81] C.B. Stunkel, "The SP2 High-Performance Switch," *IBM Systems Journal*, vol. 34, no. 2, pp. 185–204, 1995.

[82] C.B. Stunkel, D.G. Shea, and B. Abali, "The SP2 Communication Subsystem," tech. rep., IBM Thomas J. Watson Research Center, August 1994.

[83] F. May, "SP Switch Performance," tech. rep., RS/6000, SP Worldwide Product Marketing, June 1998.

[84] C. Stunkel, R. Sivaram, and D.K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact," in *ACM Annual International Symposium on Computer Architecture (ISCA'97)*, (Denver, Colorado, USA), June 1997.

[85] K.Y. Eng and M.A. Pashan, "Advances in Shared-Memory Designs for Gigabit ATM Switching," *Bell Labs Technical Journal*, pp. 175–187, 1997.

[86] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1327–1356, December 1987.

[87] M.G. Hluchyj and M.J. Karol, "Queueing in High-Performance Packet Switching," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1587–1597, December 1988.

[88] M. Galles, "Spider: A High-Speed Network Interconnect," *IEEE Micro*, pp. 34–39, January/February 1997.

[89] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "Tiny Tera: A Packet Switch Core," *IEEE Micro*, pp. 26–33, January/February 1997.

[90] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in *Proceedings Infocom'96*, pp. 296–302, 1996.

[91] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 188–201, April 1999.

[92] S. Brown et al., "The NUMAchine multiprocessor," Tech. Rep. CSRI-TR-324, CSRI, University of Toronto, 1995.

[93] W. Bux, W.E. Denzel, T. Engbersen, A. Herkersdorf, and R.P. Luijten, "Technologies and Building Blocks for Fast Packet Forwarding," *IEEE Communications Magazine*, pp. 70–77, January 2001.

[94] M.I. Irland, "Buffer Management in Packet Switch," *IEEE Transactions on Communications*, vol. COM-26, pp. 328–337, March 1978.

[95] R. Simcoe and T. Pei, "Perspectives on ATM Switch Architecture and the Influence of Traffic Patterns on Switch Design," *Computer Communications Review*, vol. 25, pp. 93–105, April 1995.

[96] C. Minkenberg, *On Packet Switch Design*. PhD thesis, Eindhoven University of Technology, September 2001.

[97] Raj Jain, *The Art of Computer Systems Performance Analysis*. New York: Wiley, 1991.

[98] John L. Hennessy and David A. Patterson, *Computer Architecture A Quantitative Approach*. San Francisco, California, USA: Morgan Kaufmann, 2nd ed., 1996.

[99] A. David E. Culler, Jaswinder P. Singh, *Parallel Computer Architecture*. Morgan Kaufmann, 1999.

[100] Craig Partridge, *Gigabit Networking*. New York: Addison-Wesley, 1993.

[101] F.T. Leighteon, *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann, 1982.

[102] D.E. Knuth, *The Art of Computer Programming*, vol. 2 Seminumerical Algorithms. Mass.: Addison-Wesley, 2 ed., 1981.

[103] D.E. Lenoski and W.-D. Weber, *Scalable Shared-Memory Multiprocessing*. San Francisco, Ca.: Morgan Kaufmann, 1995.

[104] M. Gries, *Algorithm-Architecture Trad-offs in Network Processor Design*. Tik-schriftenreihe nr.41, Swiss Federal Institute of Technology Zurich, May 2001.

[105] A. Benner, *Fibre Channel for SANs*. McGraw-Hill Telecom, 2001.

[106] M. Schwartz, *Telecommunication Networks Protocols, Modelling and Analysis*. Addison-Wesley, 1988.

[107] V. Milutinovic and M. Tomasevic, "Hardware Approaches to Cache Coherence in Shared-Memory Multiprocessors, Part 1," *IEEE Micro*, vol. 14, pp. 52–59, October 1994.

[108] V. Milutinovic and M. Tomasevic, "Hardware Approaches to Cache Coherence in Shared-Memory Multiprocessors, Part 2," *IEEE Micro*, vol. 14, pp. 61–66, December 1994.

[109] D. Graham, "SMP vs. Multi-purpose Parallel Computers," *Info DB*, vol. 10, pp. 14–29, June 1996.

[110] D. Dai and D.K. Panda, "How Can We Design Better Networks for DSM Systems," *Workshop on Parallel Computer Routing and Communication (PCRCW'97)*, June 1997.

[111] D. Dai and D.K. Panda, "How much Does Network Contention Affect Distributed Shared Memory Performance," Tech. Rep. OSU-CISRC-2/97-TR14, Dept. of Computer and Information Science, The Ohio State University, Columbus, OH 43210-1277, USA, 1997.

[112] J.I. LeBoudec, "Introduction into Computer Networking," 2001.

Index