Speculative Flow Control for High-Radix Datacenter Interconnect Routers

Cyriel Minkenberg and Mitchell Gusat

IBM Research GmbH, Zurich Research Laboratory
Säumerstrasse 4, 8803 Rüschlikon, Switzerland

Abstract

High-radix switches are desirable building blocks for large computer interconnection networks, because they are more suitable to convert chip I/O bandwidth into low latency and low cost than low-radix switches [10]. Unfortunately, most existing switch architectures do not scale well to a large number of ports. For example, the complexity of the buffered crossbar architecture scales quadratically with the number of ports. Compounded with support for long round-trip times and many virtual channels, the overall buffer requirements limit the feasibility of such switches to modest port counts. Compromising on the buffer sizing leads to a drastic increase in latency and reduction in throughput, as long as traditional credit flow control is employed at the link level. We propose a novel link-level flow control protocol that enables high-performance scalable routers based on the increasingly popular buffered crossbar architecture to scale to higher port counts without sacrificing performance. By combining credited and speculative transmission, this scheme achieves reliable delivery, low latency, and high throughput, even with crosspoint buffers that are significantly smaller than the round-trip time.

1 Introduction

The role of the interconnection network in scientific as well as commercial computer systems is of growing importance. The underlying trend is that the growing demand for computing capacity will be met through parallelism at the instruction, thread, core, and machine level.

As a consequence, the scale, speed, and efficiency of the interconnects must grow significantly, for these reasons:

- Increasing parallelism: More cores per processor, clustering, massively parallel processing.
- Computer busses are being replaced by switched networks to achieve bandwidth scalability.
- The communication infrastructure (SAN, StAN, LAN, IO, memory, cluster) is being consolidated to reduce cost and improve manageability.
- Virtualization of computing infrastructure introduces more variability (unpredictability) in the network load, potentially harming application performance, scalability, and overall efficiency. While performance remains our main concern, efficiency is becoming a particularly vexing issue in the light of strict constraints on power and heat imposed on datacenters.

As neither busses nor legacy LAN/WANs can meet all datacenter and high-performance computing requirements—notably low latency, high bandwidth, high reliability, and low cost—a number of networking technologies designed specifically for the datacenter environment have emerged. Examples are Fibre Channel, commonly employed in storage area networks (StAN), Myrinet and QsNet (Quadrics), used for low-latency, high-bandwidth inter-process communication in high-performance computing or clustering environments, and InfiniBand, designed as a comprehensive datacenter interconnect.

Our focus is the design of the switching nodes used in such networks, in particular with regard to their *radix*, i.e., the number of ports. Kim et al. show in [10] that the network latency is minimized by selecting the optimal switch radix, given the aggregate switch bandwidth feasible in current technology. This is a given value independent of the radix, i.e., the product of port count and port speed is fixed. If the radix is smaller than the optimum, the end-to-end latency increases because the hop count increases. On the other hand, if the radix is larger than the optimum, the serialization latency increases because of the reduced port speed. The result indicates that as the aggregate bandwidth increases, the optimal radix also increases. The authors estimate the optimal radices for 2003 and 2010 technologies at 40 and 127, respectively. The overall network cost also decreases as the radix increases, because for a given, fixed network bisection bandwidth, a higher radix reduces the hop count, thus requiring fewer switches and internal cables.

Modern routers are commonly based on crossbar switches of the buffered or unbuffered variety. A detailed discussion on the relative merits and drawbacks of buffered vs. unbuffered crossbars would exceed the scope of this paper. However, both scale poorly as their radix increases.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

With unbuffered crossbars the scaling bottleneck is the central scheduler: Obtaining a high-quality matching for a large number of ports in a short time slot is extremely challenging (see for instance [13]). With buffered crossbars the scaling bottleneck lies in the aggregate buffer requirements, which scale quadratically with the number of ports [7].

This scaling issue of buffered crossbars has received considerable attention recently, with the majority of solutions proposed so far focusing on architectural changes to the switch core. Instead, we can derive more substantial benefits by focusing on a router's link-level flow control (LL-FC) protocol, which governs buffer allocation and link traversal. The main contribution of this paper is a new LL-FC scheme that addresses the key requirements of computer interconnects: (1) it enables full bandwidth utilization with downstream buffers that are significantly smaller than the round-trip-time$\times$bandwidth product, (2) it reduces latency by eliminating the flow-control latency of conventional LL-FC protocols, and (3) it provides reliable delivery.

The key novel aspect is that it combines *speculative* and *credited* modes of transmission. The speculative mode predominates when utilization is low to reduce latency, whereas the credited mode predominates when utilization is high to achieve high maximum throughput. Applied to the buffered crossbar architecture, the proposed scheme can substantially reduce the overall buffer requirements.

The remainder of the paper is organized as follows. Sec. 2 reviews the role of LL-FC in interconnection networks (ICTNs) and some existing protocols. Sec. 3 explains the details of our speculative flow control (SFC) scheme. In Sec. 4 we apply SFC to the buffered crossbar architecture and show how this reduces the overall buffer requirements. In Sec. 5 we examine the performance of the proposed architecture by means of simulation. Sec. 6 discusses implementation aspects. Finally, we conclude in Sec. 7.

2 Link-Level Flow Control

Despite a large body of research results [3, 5, 9, 11, 15] accruing over the last few decades, the role of LL-FC in the design of ICTNs is often confounded with related, yet separate, topics such as routing, congestion control, and deadlock management. Nevertheless, technologies such as Fibre Channel, InfiniBand, Myrinet, or QsNet all feature some form of LL-FC. Ethernet features grant-like PAUSE frames (defined in IEEE 802.3x) that can be used to temporarily throttle the sender on a full-duplex link. We proceed with a review of the merits of LL-FC and some basic protocols.

2.1 Lossless vs. lossy networks

Interconnection networks can be categorized as follows:

- *Lossless* networks, which take every possible measure in hardware not to lose any packet, such as InfiniBand, Fibre Channel, RapidIO, PCIe, Myrinet, QsNet and others. They rely on LL-FC to prevent data loss due to buffer overflows. Transmission errors are typically recovered by a link-level retransmission mechanism implemented in hardware.
- *Best-effort* or *lossy* networks, which occasionally may drop some packets and must rely on end-to-end retries at higher layers to recover. Best known are TCP/IP and ATM networks.

The crucial difference between best-effort and lossless ICTNs is that, by design, the former allow—and even rely on—packet loss, whereas the latter will never drop a packet unless it is corrupted beyond repair. Best-effort networks typically employ end-to-end flow and congestion control in software (e.g., TCP). This reduces the complexity and cost of the network nodes, at the expense of exporting the problem from the network to the end nodes. This approach is well suited to the WAN/LAN environment, but the incurred drawbacks of reduced performance and reliability may be unacceptable for the datacenter environment. Recovery from packet drops incurs a significant latency. Moreover, the network resources already used by any packet that is dropped are wasted, including the power consumed by links and routers along the forwarding route thus far.

To prevent a source from being overly greedy and thus exacerbating this problem, TCP employs slow start with an additive-increase multiplicative-decrease (AIMD) window-adjustment algorithm. In a datacenter environment, however, the slow start is undesirable as it introduces significant latency until the full link bandwidth can be used.

Lossless networks, on the other hand, prevent buffer overflows, offer faster response time in the case of corrupted packets, do not suffer from loss-induced throughput limitations, and allow bursty flows to start sending immediately at full bandwidth. Furthermore, their goodput does not collapse when loaded beyond saturation. On the downside, LL-FC can cause congestion to propagate in multistage networks, thus inducing saturation trees.

2.2 Role of LL-FC

An LL-FC protocol provides a closed feedback loop to control the flow of data from a sender to a receiver. We distinguish various LL-FC protocols by the semantics of the control information used to inform the upstream sender about the downstream receiver's buffer status. The design of an LL-FC protocol involves trading off buffer space, bandwidth overhead, robustness, performance, and complexity.

The main objective of most LL-FC protocols is to regulate the flow of data from sender to receiver in such a way that the receiver's buffer is not forced to either drop packets for lack of space (overflow) or idle unnecessarily (underflow). More generally, LL-FC protocols fulfill one or more of the following purposes:

- Buffer overflow avoidance: Eliminating avoidable losses improves efficiency and reliability.
- Buffer underflow avoidance: Preventing buffer underruns achieves work conservation at the link level.
- Quality of Service (QoS): Per-service-class LL-FC information supports service differentiation (priorities, classes, lanes, virtual channels).
- Resource separation: By separately flow-controlling individual buffer partitions, guarantees toward deadlock avoidance or forward progress can be made. While this can be useful in conjunction with QoS support, it should be treated as an orthogonal issue.

Although LL-FC can prevent buffer overflows, packet drops due to transmission errors can never be completely avoided. Therefore, lossless networks often also employ a link-level reliable delivery (LL-RD) mechanism to ensure quick retransmission of corrupted packets. The purpose of LL-RD is to ensure error-free, in-order, single-copy delivery, typically implemented using error detection (e.g. parity or cyclic redundancy check) and, optionally, error correction (e.g., forward error-correcting code), along with a retransmission scheme (often referred to as automatic repeat request–ARQ) to retry failed packets and recover the correct packet order. Success and failure are communicated using positive (ACK) and/or negative acknowledgments (NAK), respectively. Preferably, LL-RD is implemented in hardware for optimum performance.

The most common FC protocols are *credit-*, *grant-*, and *rate-*based FC. In grant-based FC, also referred to as on-off FC, the semantics of an FC message are either "stop sending" or "start sending". In credit-based FC, the semantics are "permission to send up to x data units" in the case of absolute credits, or "permission to send up to x additional data units" in the case of relative (incremental) credits. Finally, the semantics of rate-based FC are either "send slower (by a factor x)" or "send faster (by a factor x)."

The *round-trip time* (RTT) is the sum of the latencies δ_d and δ_c of the forward (data) path and the reverse (FC) path, respectively: RTT $= \delta_d + \delta_c$. The RTT is often normalized with respect to the packet duration. The normalized round trip equals $\tau = \left\lceil \frac{B_d W_d \text{RTT}}{L_d} \right\rceil$, where B_d is the speed of the data channel, W_d the width of the data channel, and L_d the length of a data packet. The RTT plays a crucial role in properly dimensioning the receiver buffer size $Q_{\mathbf{R}}$ to achieve losslessness and work conservation.

2.3 Credit FC

Credit-based FC [9, 11, 15] typically uses relative (incremental) credit updates for efficiency, and requires one round trip worth of buffer space ($Q_{\mathbf{R}} = \tau$) to achieve losslessness and work conservation, whereas grant-based FC requires two RTs worth of buffer space ($Q_{\mathbf{R}} = 2\tau$) to do the same. However, relative credit FC is a stateful protocol, meaning that the current state as observed by the sender depends on the history of FC information. On/off grant FC, on the other hand, is stateless, meaning that the current state depends only on the most recent FC information. Hence, credits are more efficient, whereas grants are more robust.

We briefly review relative credit FC, considering a simple point-to-point scenario with one sender **S** and one receiver **R** communicating over a bidirectional link. Both **S** and **R** have buffers to store packets; we assume fixed-size packets (cells), although this is not a strict requirement. Initially, **R** advertises the number of credits to **S**—we assume a granularity of one cell per credit, although this may vary per implementation. **S** maintains a credit count that tracks the difference between the aggregate number of credits received and the number of credits consumed. **S** decrements the credit count for every cell sent; when the count reaches zero, no more cells can be sent until new credits arrive. **R** issues a new credit every time a cell departs from its buffer.

The ratio between the normalized RTT and the receiver's buffer size $Q_{\mathbf{R}}$ (in cells) determines the maximum link utilization ρ_{max} as follows: $\rho_{\text{max}} = \min\left(\frac{Q_{\mathbf{R}}}{\tau}, 1\right)$.

Consider the minimum turn-around time of a specific credit; if the credit is issued at t_0, it arrives at the sender at $t_0 + \delta_c$. Assuming the sender has traffic queued up at that time and immediately sends a cell consuming the credit, the corresponding storage location will be reused at $t_0 + \delta_c + \delta_d$ at the earliest, so it follows that the turn-around time equals RTT. Moreover, as there are $Q_{\mathbf{R}}$ credits, a maximum of $Q_{\mathbf{R}}$ cells can be injected during one RTT, hence the maximum injection rate equals $\min\left(\frac{Q_{\mathbf{R}}}{\tau}, 1\right)$. This implies that $Q_{\mathbf{R}} \geq \tau$ must hold to achieve full link utilization.

Consider the situation where, at the start of time slot t_1, the receiver buffer is completely full, so that no free credits are in circulation. During t_1, the receiver removes one cell from its buffer and issues a credit. The *restart* time is the latency from t_1 until the sender can resume transmission. It follows that the restart time of credit FC equals δ_c. It can be shown that the worst-case restart time of grant FC equals $\tau + \delta_c$.

A major difference between credit and grant FC is that credit FC operates losslessly with any buffer size greater than zero, whereas grant FC requires at least one RTT worth of overflow buffer space. However, performance suffers if the downstream buffer size is less than τ cells. As Fig. 1 illustrates, in this case the RTT becomes a multiplicative factor in the burst latency. Here, a burst of six cells is transmitted through a crosspoint that has room for only two cells (i.e., two credits). Between every pair of two cells, a latency of one full RTT is incurred. Hence, the end-to-end latency of the burst equals three times the RTT.

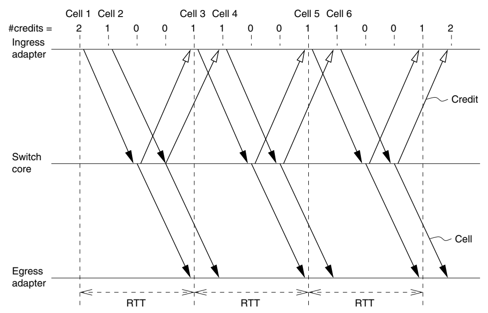

Figure 1. Burst latency. Q_{R} = 2 cells, burst size = 6 cells, RTT = 4 cell slots.

2.4 ACK/NAK FC

A fourth, rarely used FC protocol is known as *ACK/NAK* FC [4, Sec. 13.3.3]. With this protocol, the sender does not maintain any state information about the receiver buffer. The sender is free to send whenever it has data. When a cell arrives at the receiver, it gets stored if there is room or dropped if there is none. In the former case the cell is acknowledged positively (ACK), otherwise it is acknowledged negatively (NAK). The sender stores unacknowledged cells in a retransmission buffer, and the receiver must ensure that cells are reordered properly, because cell drops lead to out-of-order arrivals.

This type of FC is rather unpopular because it is wasteful of bandwidth, while having similar overall buffer requirements as credit FC. However, it does have two significant benefits. First, the maximum link utilization does not depend on the ratio $Q_{\mathbf{R}}/\tau$. ACK/NAK FC can operate at full link bandwidth with $Q_{\mathbf{R}} = 1$ regardless of τ, as long as the receiver buffer is not full. Second, the restart time is equal to zero, because the condition to start transmitting does not depend on the state of the receiver buffer. Because reception is not guaranteed, the sender has a retransmission buffer of size Q_{RTX} to store cells that are waiting to be acknowledged. The sender can only transmit when the retransmission buffer is not full. It follows that to obtain full link utilization $Q_{\text{RTX}} \geq \tau$ must hold. Therefore, the overall minimum buffer requirement equals $Q_{\mathbf{R}} + Q_{\text{RTX}} = 1 + \tau$.

2.5 Controlling multiple downstream buffers

So far, we have only considered the case with a single downstream buffer. However, in many applications, the receiver may have multiple separate buffers, each being flow-controlled independently. A common example is a link that supports multiple, say L, virtual channels. To ensure that each lane can make progress independently of the other channels, each lane has a private buffer space that is managed with its own credits. Another highly relevant example is the $N \times N$ buffered crossbar architecture (see also Sec. 4), which performs switching among N inputs and N outputs. This architecture features N separate buffers at the receiving end of each input link.

In the general case with M downstream buffers, the overall buffer sizes under the requirement that each buffer must be able to sustain full link bandwidth scale as $M\tau$ for credit FC (M receiver buffers) and $\tau + M$ for ACK/NAK FC (one retransmission buffer and M downstream buffers). Clearly, the latter scales much better than the former. This advantage derives from the fact that a single retransmission buffer is sufficient because there are only τ cells in flight (one link) regardless of the number of downstream buffers.

3 Speculative Flow Control

We propose a novel LL-FC protocol that combines the properties of credits and ACK/NAK FC in such a way that it shares the advantages of both. Because of its speculative aspect, we refer to it as *speculative flow control* (SFC). SFC is a point-to-point protocol in which a sender and a receiver communicate over a bidirectional channel. For the purpose of the discussion, the downstream channel (sender to receiver) carries data messages, i.e., cells, and the upstream channel carries FC messages. The receiver may have multiple (M), independently managed buffers in which cells can be stored. These buffers may correspond to, e.g., virtual channels or to different output ports. We assume that the sender maintains one queue corresponding to every downstream buffer to prevent head-of-line blocking. The receiver issues separate credits for every buffer, i.e., every credit is uniquely associated with one buffer. Correspondingly, the sender maintains an available credit count for every buffer.

3.1 Rules of operation

SFC combines credited and speculative modes of operation, with the credited mode taking precedence over the speculative one. In the speculative mode, the sender is allowed to transmit cells even without sufficient credits. The following rules govern the operation of the SFC protocol:

$\mathcal{R}1$ A cell is eligible for credited transmission if the sender has sufficient credits for the cell's destination downstream buffer, i.e., the available credits represent at least as much buffer space as is needed to store the entire cell. The sender may perform a speculative transmission in a given time slot if and only if no cell is eligible for credited transmission. A transmitted cell is referred to as speculative if there were insufficient credits, otherwise it is referred to as credited.

$\mathcal{R}2$ The receiver drops an incoming cell if its buffer is full or the cell is corrupted, out of order, or a duplicate. For every dropped cell except for duplicates, a NAK identifying the dropped cell is returned. To this end, every cell carries a sequence number.

$\mathcal{R}3$ The receiver issues an ACK for a cell at the instant the cell is removed from its buffer, i.e., when the corresponding buffer space is freed up.

$\mathcal{R}4$ Each cell remains stored at the sender until it is positively acknowledged.

$\mathcal{R}5$ Each cell may be speculatively transmitted at most once. All retransmissions must be performed as credited transmissions.

$\mathcal{R}6$ The sender consumes credit for every cell sent, i.e., for speculative as well as credited transmissions.

$\mathcal{R}7$ The cells at the sender can be partitioned into unsent cells and cells waiting for positive acknowledgment. The latter group can be partitioned again along two orthogonal lines: They are either unacknowledged or negatively acknowledged and they are either speculative or credited. When credits are available, speculative cells and negatively acknowledged cells take precedence over unsent cells, which in turn take precedence over unacknowledged credited cells. The latter are only eligible for service after a certain expiry time to deal with corrupted cells and ACKs.

In the remainder of this section, we will explain the rationale behind these rules.

$\mathcal{R}1$ implies that credited transmissions take strict precedence over speculative ones. This ensures that the desirable properties, specifically high utilization, of credits are preserved.

The receiver drops a cell if either the buffer is full (speculative cells only), the cell arrives out of order (OOO), or an uncorrectable transmission error corrupted the cell. In all of these cases, $\mathcal{R}2$ requires that the receiver returns a NAK for the dropped cell. This NAK comprises a sequence number uniquely identifying the cell. The receiver does *not* send NAKs in response to duplicately delivered cells because there is no need for a retransmission; these duplicate cells are discarded silently. In an alternative implementation, the receiver could accept OOO cells and perform resequencing on a per-flow basis to ensure in-order delivery. This reduces the number of retransmissions at the cost of resequencing buffers and logic.

Instead of being communicated explicitly, credits are conveyed implicitly by the acknowledgments. According to $\mathcal{R}3$ the receiver returns a positive acknowledgment (ACK) when a cell *leaves*, not when it enters, its buffer. Hence, an ACK is equivalent to an incremental credit, whereas a NAK is equivalent to an incremental credit for a dropped cell. From the perspective of LL-RD, the ACK can be returned upon entrance, but by delaying it until exit, the overhead of LL-FC and LL-RD can be condensed into a single FC message. Moreover, this approach enables cell replacements, as discussed in Sec. 3.2. The main drawback is that cells remain stored longer at the sender, so it impacts the retransmission buffer dimensioning. NAKs, on the other hand, are returned right after cell arrival.

According to $\mathcal{R}4$, all cells remain stored in the sender's buffer until positively acknowledged. This holds for speculative as well as credited cells, even though credited cells will not be dropped as a result of overflow. However, the RD mechanism also covers other sources of cell loss, such as unrecoverable transmission errors. Hence, SFC fully integrates LL-RD, dealing with protocol-specific as well as protocol-independent causes of loss.

As excessive speculation may cause bandwidth wastage, $\mathcal{R}5$ limits the number of speculations to at most one per cell. Also, cells waiting for acknowledgement and negatively acknowledged cells are not eligible for speculation.

$\mathcal{R}6$ specifies a *conservative credit* policy. This means that a credit is consumed even though there may be no space in the receiving buffer. Correspondingly, the receiver returns a credit for every incoming cell, even the ones that have been dropped. $\mathcal{R}6$ implies that the credit count may be negative. The classification into speculative and credited cells ($\mathcal{R}1$) can be reformulated as follows: A cell selected for transmission is *credited* if the corresponding credit count is greater than or equal to zero after decrementing, and otherwise it is a *speculative* cell. Any queue for which the credit count is less than or equal to zero (before decrementing) can only perform speculative transmissions. The benefits of adopting $\mathcal{R}6$ are that all credited cells are guaranteed to not be dropped because of buffer overflow and that cells need not be explicitly marked as credited or speculative.

The purpose of $\mathcal{R}7$ is to expedite the reliable delivery of speculative cells, which have a relatively high probability of loss. As credited cells can only be dropped because of physical-level errors (which are assumed to be rare) or being OOO, they receive the lowest priority. To ensure that unacknowledged cells do not wait forever when a severe transmission error damages the cell so badly that the receiver does not even detect its presence or when the ACK/NAK is lost, cells should be aged. Cells having an age over a certain (conservative) threshold should be considered as negatively acknowledged and treated as such.

The specific policies used to arbitrate among eligible cells for either speculative or credited transmission can be chosen according to preference. The only restriction is that cells belonging to the same flow should be served in the order of their arrival.

3.2 SFC with replacement

The SFC rules of operation specify that a cell should be stored at the sender until acknowledged. This opens up the possibility of *replacing* cells already stored at the receiver by newly arriving ones. Such replacements are useful in several contexts, for example to support multiple levels of priority. Using conventional credits, dedicated receiver buffer space must be allocated to every priority to ensure progress and prevent *priority inversion* [12]. This occurs when low-priority cells hog the receiving buffer, thus preventing higher-priority ones from overtaking them. With SFC, it is possible to *replace* a low-priority cell when a high-priority one arrives at a full buffer. To signal the replacement, the receiver issues a NAK for the replaced (dropped) cell.

However, this mechanism only works properly when the sender's retransmission buffer is larger than the downstream buffer. Otherwise, injections of new cells would be prevented by a full retransmission buffer. Hence, the "oversize" of the retransmission buffer directly determines how many replacements can be made. Alternatively, the replaced cells could be removed from the retransmission buffer and requeued in the regular sender buffer. In practice, this would be difficult to implement in hardware because of expensive dequeue operations at arbitrary points in the retransmission buffer. This also creates sequencing issues and violates the credit semantics if the sender buffer is also a receiver buffer (e.g. in multistage networks).

3.3 Applications

A few potential fields of application for SFC are:

- *Datacenter interconnection networks:* SFC enables a mix of lossless and lossy traffic to coexist in the same network. Furthermore, it simultaneously offers high reliability for storage and system area networks, low latency for parallel computing applications, and high utilization.
- *Buffered crossbar switches:* The buffer requirements of this type of switch scale quadratically with the port count. Applying SFC may reduce this requirement. We study this application in detail in Sec. 5.
- *Quality-of-service:* Providing QoS using conventional credits requires pre-allocating buffer space to every traffic class. With SFC, the buffer space can be shared dynamically among multiple priorities, without leading to priority inversion (Sec. 3.2), which occurs when low-priority traffic is stalled in a shared buffer, thus preventing higher-priority traffic from entering.
- *Congestion control:* The problem of downstream buffer hogging is also responsible for *tree saturation* [16] in lossless interconnects: Cells destined for a "hot" (i.e., overloaded) node use up a disproportionate share of network buffers. In severe cases, this can lead to a collapse of aggregate network throughput. As SFC enables replacement of cells in downstream buffers, it may be useful for fighting tree saturation. This application is left for future study.

4 An SFC-enabled Buffered Crossbar Switch

Having studied the role of LL-FC in interconnection networks and defined the SFC protocol, we now proceed with

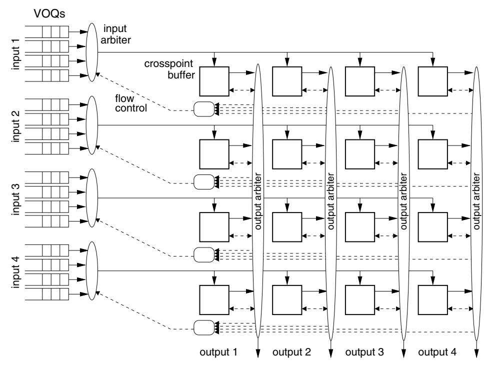

Figure 2. Combined- input- and crosspoint-queued switch, $N = 4$.

a specific SFC case study, namely, the combined-input-crosspoint-queued (CICQ) switch architecture (shown in Fig. 2), which features upstream buffers (input queues) organized as virtual output queues (VOQs) and an $N \times N$ buffered crossbar switching core; the flow control protocol is usually based on incremental credits.

Shared-memory switch architectures have been unable to keep pace with the rising port counts and line rates, because implementing a shared memory with the required aggregate throughput is extremely difficult, the main limitation being wiring rather than logic gates. The buffered crossbar has emerged as a viable alternative to the shared-memory architecture. By dedicating a buffer to every input-output combination, all memories operate at one instead of N times the line rate.

4.1 Related work

Packet switches based on buffered crossbars have attracted increasing attention [1, 8, 10, 14, 18, 21] in recent years. The advantages of the CICQ architecture are that it has a balanced distribution of scheduling complexity, memories that operate no faster than the line rate, and excellent performance characteristics under a wide range of traffic patterns *without* requiring speedup. Its main drawback is that its memory requirements grow quadratically with N. When taking into account the (normalized) round trip τ between the line cards and the buffered crossbar as well as support for P priorities (or virtual channels), the overall buffer size scales as $O(N^2 P\tau)$. This linear dependency on the RTT is a direct consequence of credit FC, as explained in Sec. 2.3. As the switch radix N increases, this memory requirement quickly becomes prohibitively large. Despite ever-increasing CMOS densities, the amount of buffering available per crosspoint remains small. The key issue is that there are N^2 individual memories, each with their own (significant) control overhead. Several recent studies have aimed at reducing the buffer requirements:

In [10], a hierarchical crossbar architecture is proposed that partitions the ports into groups of n, such that there are $(N/n)^2$ subswitches, each having buffers only at its in- and outputs, but no internal buffers. This reduces the overall buffer area by a factor n to $O(N^2 P\tau/n)$, with $1 < n < N$.

The architecture described in [2] collapses all crosspoint buffers belonging to the same output into one small output queue that is managed with credits that are allocated on demand to specific inputs using a request-grant protocol. Using this approach, the overall buffer area scales as $O(NP)$. This approach has three major drawbacks: (1) the minimum latency increases by one additional RTT, (2) the write bandwidth of the output buffers is higher than the line rate (it is basically an output-queued switch), and (3) the output ports are assumed to never be flow-controlled.

In [17], the authors propose an architecture that features a load-balancing stage in front of the buffered crossbar. Cells for a given output can be stored in any of the crosspoints associated with the destination output, which enables sharing of the available buffer space and hence reduces the overall buffer area by a factor N to $O(NP\tau)$. However, because multiple inputs may try to access the same crosspoint buffer simultaneously, this approach also requires a request-grant protocol and a bipartite graph-matching algorithm to resolve contention. This adds significant latency and complexity.

In [20], the τ factor is addressed by implementing FIFO buffers close to the buffered crossbar to cope with the RTT. The overall buffer area scales as $O(NP(N + \tau))$. Unfortunately, this induces head-of-line blocking.

In [19] a rate-controlled CICQ switch is proposed. Instead of employing an LL-FC protocol, it exchanges VOQ occupancy information among all input queues to determine suitable VOQ service rates. Using this approach, each crosspoint scales by N rather than τ, so the overall buffer area scales as $O(N^3 P)$. Hence, this approach only makes sense if $\tau > N$. Moreover, it incurs significant worst-case latency because of the global VOQ state exchange process. As in [2], this approach assumes that the output ports are never flow-controlled. Both schemes may lead to buffer overflows when the outputs cannot always be served.

Four of these schemes achieve their buffer area reduction by modifying the switch architecture while keeping the basic credit FC in place, whereas the last one [19] omits LL-FC altogether.

4.2 Reducing buffer area by SFC

We propose to tackle the buffer area issue by modifying the LL-FC protocol rather than the architecture. The SFC protocol is very well suited to buffered crossbars, because there are N downstream buffers per input link, as described in Sec. 2.5. Hence, we can expect to gain a significant advantage from SFC. In general, the retransmission buffer size Q_{RTX} should be dimensioned as follows: $Q_{\text{RTX}} = \max(\tau, Q_{\mathbf{R}} N)$. This ensures that the retransmission buffer can store at least a full round trip worth of cells *or* all cells in the crossbar row, whichever is larger. In general, the crosspoint buffer size $Q_{\mathbf{R}}$ should be sized such that the total buffer space available per row is at least τ cells. This ensures that the link can be fully utilized under high load. Under this condition, Q_{RTX} evaluates to $Q_{\mathbf{R}} N$. In the ideal case, this implies $Q_{\mathbf{R}} = \tau/N$, so the total buffer space needed for a CICQ with SFC equals $NQ_{\mathbf{R}} + Q_{\text{RTX}} = 2Q_{\mathbf{R}}N = 2\tau$ cells per row, or $2N\tau$ in total. As the total buffer space for a CICQ with conventional credit FC equals $N^2\tau$ cells, the aggregate buffer requirement is reduced by a factor of $N/2$. In addition, half of the memory required for SFC resides on the line cards, where resources are generally less scarce and less expensive than in the core.

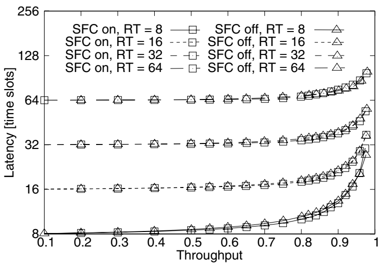

Figure 3. Latency–throughput characteristics with uniform Bernoulli traffic. $\tau = 8, 16, 32, 64$ **time slots.** $N = 16$**, crosspoint memory size** $Q_{\mathbf{R}} = \lfloor \tau/N \rfloor + 2$**, window size** $Q_{\text{RTX}} = Q_{\mathbf{R}} N$**.**

5 Simulation results

We built a software model of the proposed architecture with the OMNeT++ simulation environment to obtain its performance characteristics by simulation. Specifically, we measure the mean aggregate throughput and the mean latency. The latency is measured per burst, i.e., from the time the first cell of a burst enters the system until the last cell exits. The results were obtained using the Akaroa2 environment with a statistical confidence of at least 95% and a precision of 0.4% on the throughput and 5% on the delay.

In our experiments, we study the effect of SFC on the performance of a CICQ switch with $N = 16$ and $N = 64$ ports. The arbitration policies at the VOQs and the output queues are both longest queue first (LQF). We vary the RTT and set the crosspoint buffer size to $\lfloor \tau/N \rfloor + 2$ cells.[1] We use Bernoulli and bursty arrivals with an average burst size of 10 and 20 cells/burst. We measure the latency–throughput characteristics with a uniform destination dis-

[1]Because of model implementation details, at least two cells per crosspoint are required.

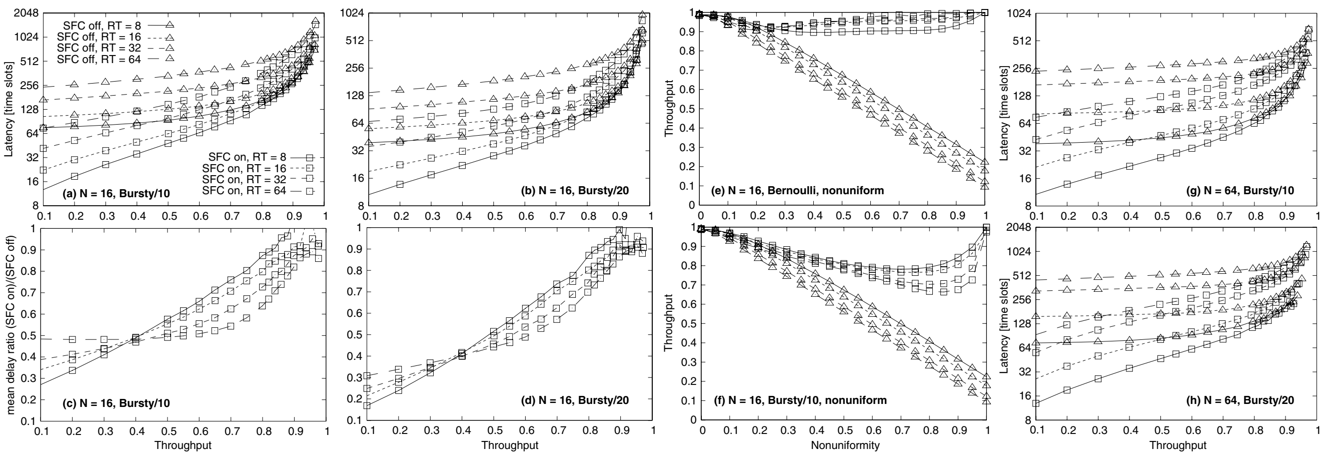

Figure 4. Latency–throughput characteristics with uniform and nonuniform traffic of varying burstiness. $\tau = 8, 16, 32, 64$ **time slots. Crosspoint memory size** $Q_{\mathbf{R}} = \lfloor \tau/N \rfloor + 2$**, window size** $Q_{\text{RTX}} = Q_{\mathbf{R}}N$**.**

tribution and the throughput–nonuniformity characteristics with a nonuniform destination distribution.

Figure 3 shows the results for uniform Bernoulli traffic. In this case, enabling SFC makes no observable difference, regardless of RTT. Because (a) there is enough buffer space to cover one RTT, (b) the traffic is uniform, and (c) every burst comprises just a single cell, the likelihood that there is no opportunity for a credited transmission is very low; hence, the effect of speculation is negligible.

Figures 4(a,b) show the latency–throughput results for bursty traffic, whereas Figs. 4(c,d) show the latency improvement ratios obtained by using SFC, i.e., the latency with SFC divided by the latency without SFC. We observe a notable latency reduction, especially at low loads. Moreover, this difference increases with the RTT. As the crosspoint size is smaller than the average burst size, the tail of a burst must often wait in the line card until the head of the burst has exited the crossbar. This has the undesirable effect of making the mean latency dependent on the RTT, which is clearly visible in Figs. 4(a,b). The ratio between the mean burst size B and crosspoint size $Q_{\mathbf{R}}$ directly determines the latency at very low loads (10%). The ratio $\left\lceil \frac{B}{Q_{\mathbf{R}}} \right\rceil$ evaluates to 5 ($Q_{\mathbf{R}} = 2$), 4 ($Q_{\mathbf{R}} = 3$), 3 ($Q_{\mathbf{R}} = 4$), and 2 ($Q_{\mathbf{R}} = 6$), respectively. These values correspond quite closely to the ratio (RTT-multiple) between the mean latencies at 10% of the "SFC off" curves in Fig. 4(a) and the RTT. The cause of this high latency was illustrated earlier in Fig. 1.

This effect becomes more pronounced as the burst size increases, as evidenced by Fig. 4(b), which shows results for bursty traffic with an average burst size of 20 cells/burst. The RTT-multiples at low load are around 10, 7, 5, and 4, respectively, which again correspond to the ratio $\left\lceil \frac{B}{Q_{\mathbf{R}}} \right\rceil$. These results demonstrate that when τ is much larger than the crosspoint size, using SFC can reduce the average burst latency by a factor up to $\left\lceil \frac{B}{Q_{\mathbf{R}}} \right\rceil$.

The behaviour of SFC and non-SFC converges as the load approaches 100%. This confirms that the interaction of speculative and credited operation automatically adjusts to the load level, with speculations predominating at low loads, and credited transmissions prevailing at high loads.

Figures 4(e,f) show the maximum throughput as a function of the nonuniformity factor w. The drastic difference between operation with and without SFC is due to the ratio between τ and crosspoint size $Q_{\mathbf{R}}$: the maximum throughput without SFC is limited to $\min(Q_{\mathbf{R}}/\tau, 1)$ when $w = 1$, whereas SFC enables full throughput under unbalanced traffic, owing to its speculative mode of operation. This illustrates that SFC can also be effective at high loads; in general, SFC is effective when the achievable throughput for a given crosspoint exceeds the rate supported by the available credits. For example, this is the case with bursty uniform traffic at low to medium loads or with strongly nonuniform traffic.

The results also demonstrate that the difference in performance increases with N. For a given τ, as N increases the required crosspoint buffer size $Q_{\mathbf{R}}$ decreases because there are more crosspoint buffers per row. As a result, the latency ratio $\left\lceil \frac{B}{Q_{\mathbf{R}}} \right\rceil$ increases. In other words, the negative effect on latency of small crosspoint buffers is exacerbated as the switch radix increases, reinforcing the applicability of SFC to high-radix computer interconnect switches. This behavior is evident when comparing Figs. 4(g,h), which show the results for N = 64, burst sizes 10 and 20, and τ = 8, 16, 32, and 64 time slots, with Figs. 4(a,b). The latency reduction owing to SFC is even more pronounced in this case.

We have also studied the behavior with two traffic

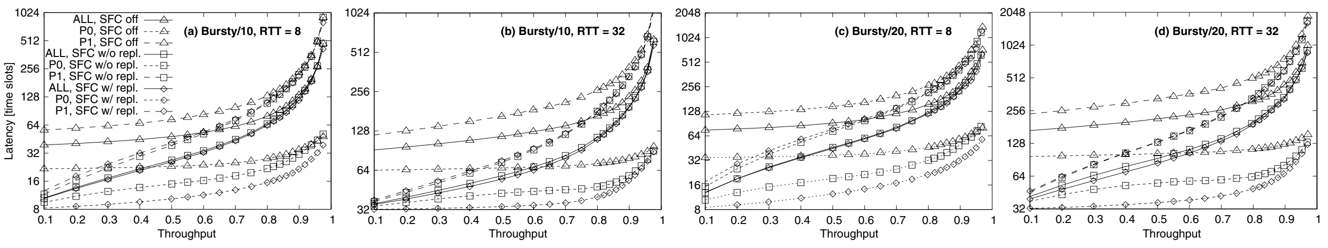

Figure 5. Latency–throughput characteristics with two priorities. Uniform traffic of varying burstiness. $\tau = 8, 16, 32, 64$ time slots. $N = 16$, crosspoint memory size $Q_{\mathbf{R}} = \lfloor \tau/N \rfloor + 2$, window size $Q_{\text{RTX}} = Q_{\mathbf{R}}N$.

classes. As an example, we consider strict priority scheduling, in which the higher priority always takes precedence. Figure 5 shows the results for bursty traffic (mean burst size 10 and 20 cells), with τ = 8 and 32 time slots. Each subfigure shows the delay vs. aggregate throughput characteristics of priority 0 (P0, high), priority one (P1, low), and aggregate (ALL), comparing the same system (a) without SFC, (b) with SFC but without replacement, and (c) with SFC *and* replacement. The replacement policy is activated when an arriving cell finds the buffer full. If there are any cells of lower priority, the incoming cell replaces the most recent arrival of the lowest priority present in the buffer. In principle, replacement can be applied to any QoS mechanism, although the implementation complexity of the replacement policy may be prohibitive.

First, speculation drastically reduces the latency of low- as well as of high-priority traffic. With replacement, the high-priority latency is further reduced. This gain does not incur an increase of the mean latency of low-priority traffic.

6 Implementation Issues

6.1 Overhead

SFC introduces some overhead on the upstream as well the downstream channel. On the downstream channel, every cell must carry a sequence number uniquely identifying it within the current transmission window of the sender. As the maximum window size equals Q_{RTX}, the sequence number should be $\log_2(Q_{\text{RTX}})$ bits long (assuming go-back-N). The ACKs/NAKs that flow on the upstream path consist of an ACK/NAK flag (1 bit), a buffer (crosspoint) identifier ($\log_2(N)$ bits), a sequence number ($\log_2(Q_{\text{RTX}})$ bits), and (optionally) a credit count (when using aggregated ACKs). In a practical implementation, the FC information could be piggybacked on data traffic travelling upstream, e.g. embedded in the cell headers.

In a buffered crossbar switch, another concern is the number of ACK/NAKs to convey per time slot [6]. In principle, up to N cells can depart from the same crossbar row. To reduce the required control channel bandwidth, our model allows sending one NAK plus at most one ACK per time slot. As this can lead to an accumulation of ACKs (but not of NAKs), the crossbar must maintain a queue of to-be-returned ACKs for every row. However, this issue is common to all LL-FC schemes; when multiple events (arrivals or departures) occur at once in the same row, multiple FC events may be triggered.

6.2 Hardware

The main hardware complexity burden of SFC rests on the sender. First, a retransmission buffer is required with an accompanying queuing structure. New entries are always enqueued at the tail of the queue, but reads and dequeue operations may occur at any point in the queue. Processing an ACK/NAK involves finding the entry with the corresponding sequence number (and buffer ID) in the retransmission queue. Upon ACK, the corresponding queue entry (and any preceding ones) is removed. Upon NAK, the entry is marked so that it becomes eligible for retransmission.

For every incoming cell, the receiver must determine whether there is room in the destination buffer, and whether the cell contains any errors, is in the correct order, or is a duplicate. If there is an error, the cell is out of order, or the buffer is full, the cell is dropped and a NAK returned. If the cell is a duplicate, it is dropped without sending a NAK. Otherwise, the cell is stored and the expected next sequence number is incremented. The receiver issues an ACK for a cell when it is removed from the buffer.

In the case of multiple downstream buffers, the number of ACK/NAKs per time slot (as described above) also determines the rate of operations on the retransmission queue.

To minimize the number of retransmissions caused by out-of-order arrivals, the sender keeps track of the sequence number of the last negatively acknowledged cell for every downstream buffer. Further speculative transmissions to this buffer are not allowed until that cell is positively acknowledged. This reduces bandwidth wastage due to long trains of OOO cells following an unsuccessful speculation. Once the sender knows a cell was lost, further speculative transmissions are useless as these are sure to be dropped. To reduce the time during which no speculative transmissions are allowed, this sequence number can be reset as soon as the corresponding cell is sent credited instead of waiting until the ACK is received, because the likelihood of a credited cell being dropped is very low.

7 Conclusions

Motivated by the need for high-radix switches for computer interconnection networks, we proposed a novel link-level flow-control method: speculative flow control (SFC). It specifically addresses the ICTN requirements of losslessness, low error rates, and low latency. SFC combines credited and speculative modes of transmission, in which the credited mode corresponds to conventional credit flow control, whereas the speculative mode allows "violations" of the credit semantics. This allows significant reductions in latency for bursty traffic. Furthermore, SFC impacts buffer sizing in two ways. First, buffers that traditionally needed to be allocated (partitioned) on a per-flow basis can now be shared, leading to substantial buffer size reductions. Second, buffers that needed to be dimensioned proportional to the RTT to achieve full link utilization can now be significantly reduced.

In the specific application to buffered crossbar switches, SFC enables a drastic reduction of the buffer requirements by relaxing the need to provide one full RTT worth of buffer space per crosspoint. Our results show that the overall buffer size can be reduced by a factor of $N/2$, independent of the RTT, while maintaining full system performance.

Acknowledgments

The authors are grateful to José Duato for discussions on an early version of SFC, and to Ton Engbersen for his insights and careful reviews of our work.

References

[1] F. Abel, C. Minkenberg, R. Luijten, M. Gusat, and I. Iliadis. A four-terabit packet switch supporting long round-trip times. *IEEE Micro*, 23(1):10–24, Jan./Feb. 2003.

[2] N. Chrysos and M. Katevenis. Scheduling in switches with small internal buffers. In *Proc. IEEE GLOBECOM 2005*, volume 1, pages 614–619, St. Louis, MO, Nov. 28–Dec. 2 2005.

[3] W. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Syst.*, 1(3):187–196, Oct. 1992.

[4] W. J. Dally and B. P. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, San Francisco, CA, 2003.

[5] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Syst.*, 4(12):1320–1331, Dec. 1993.

[6] F. Gramsamer, M. Gusat, and R. Luijten. Flow control scheduling. *Elsevier J. Microprocessors and Microsystems*, 27(5–6):233–241, June 2003.

[7] M. Gusat, F. Abel, F. Gramsamer, R. Luijten, C. Minkenberg, and M. Verhappen. Stability degree of switches with finite buffers and non-negligible round-trip time. *Elsevier J. Microprocessors and Microsystems*, 27(5–6):243–252, June 2003.

[8] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos. Variable packet size buffered crossbar (CICQ) switches. In *Proc. IEEE International Conference on Communications (ICC 2004)*, pages 1090–1096, Paris, France, June 20–24 2004.

[9] M. Katevenis, D. Serpanos, and P. Vatsolaki. ATLAS I: A general-purpose, single-chip ATM switch with credit-based flow control. In *Proc. IEEE Hot Interconnects IV Symposium*, pages 63–73, Stanford, CA, Aug. 15–17 1996.

[10] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta. Microarchitecture of a high-radix router. In *Proc. ISCA 2005*, pages 420–431, Madison, WI, June 2005.

[11] H. Kung, T. Blackwell, and A. Chapman. Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing. In *Proc. ACM SIGCOMM*, pages 101–114, London, UK, Aug. 31–Sept. 2 1994.

[12] R. Luijten, C. Minkenberg, and M. Gusat. Reducing memory size in buffered crossbars with large internal flow control latency. In *Proc. IEEE GLOBECOM 2003*, volume 7, pages 3683–3687, San Fransisco, CA, Dec. 1–5 2003.

[13] C. Minkenberg, F. Abel, P. Müller, R. Krishnamurthy, and M. Gusat. Control path implementation of a low-latency optical HPC switch. In *Proc. Hot Interconnects 13*, pages 29–35, Stanford, CA, Aug. 17–19 2005.

[14] M. Nabeshima. Input-queued switches using two schedulers in parallel. *IEICE Trans. Commun.*, E85-B(2):523–531, Feb. 2002.

[15] C. Özveren, R. Simcoe, and G. Varghese. Reliable and efficient hop-by-hop flow control. *IEEE J. Sel. Areas Commun.*, 13(4):642–650, May 1993.

[16] G. Pfister and V. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Trans. Computers*, C-34(10):933–938, Oct. 1985.

[17] R. Rojas-Cessa, Z. Dong, and Z. Guo. Load-balanced combined input-crosspoint buffered packet switch and long round-trip times. *IEEE Commun. Lett.*, 4(7):661–663, July 2005.

[18] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Chao. CIXB-1: Combined input-one-cell-crosspoint buffered switch. In *Proc. 2001 IEEE Workshop on High-Performance Switching and Routing HPSR 2001*, pages 324–329, Dallas, TX, May 2001.

[19] K. Yoshigoe. Rate-based flow-control for the CICQ switch. In *Proc. 30th IEEE Conference on Local Computer Networks (LCN 2005)*, pages 44–50, Sydney, Australia, Nov. 15–17 2005.

[20] K. Yoshigoe. The CICQ switch with virtual crosspoint queues for large RTT. In *Proc. IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, June 11–15 2006.

[21] K. Yoshigoe, K. Christensen, and A. Jacob. The RR/RR CICQ switch: Hardware design for 10-Gbps link speed. In *Proc. IEEE Int. Performance, Computing, and Communications Conf. IPCCC 2003*, pages 481–485, Phoenix, AZ, Apr. 9–11 2003.