

---

# DESIGNING A CROSSBAR SCHEDULER FOR HPC APPLICATIONS

---

**Cyriel Minkenberg**

**François Abel**

**Peter Müller**

**Raj Krishnamurthy**

**Mitchell Gusat**

**Peter Dill**

**Ilias Iliadis**

**Ronald Luijten**

**IBM Zurich Research**

**Laboratory**

**B. Roe Hemenway**

**Richard Grzybowski**

**Corning Inc.**

**Enrico Schiattarella**

**Politecnico di Torino**

A CRUCIAL PART OF ANY HIGH-PERFORMANCE COMPUTING (HPC) SYSTEM IS ITS INTERCONNECTION NETWORK. CORNING AND IBM ARE JOINTLY DEVELOPING A DEMONSTRATION INTERCONNECT BASED ON OPTICAL CELL SWITCHING WITH ELECTRONIC CONTROL. KEY INNOVATIONS IN THE SCHEDULER ARCHITECTURE DIRECTLY ADDRESS THE MAIN HPC REQUIREMENTS: LOW LATENCY, HIGH THROUGHPUT, EFFICIENT MULTICAST SUPPORT, AND HIGH RELIABILITY.

..... Massively parallel high-performance computing (HPC) systems require large-scale, high-bandwidth, low-latency interconnection networks. The Corning-IBM joint Optical Shared Memory Supercomputer Interconnect System (Osmosis) project explores the opportunity to advance the role of optical-switching technologies in such systems.<sup>1</sup> There are two strong reasons for preferring an optical interconnect over an electrical one:

- *Distance.* High-end HPC systems—supercomputers—are typically very large and highly distributed, with diameters spanning tens to hundreds of meters, so the capability of covering such distances with high bandwidth and low latency is a significant advantage. For transmitting high data rates over long distances, fiber is far better suited than copper. Also, optical interconnects neither generate nor are sensitive to electromagnetic interfer-

ence—a clear benefit at multigigabit rates.

- *Power.* An optical switch typically consumes significantly less power than a comparable electronic switch does, because its power consumption is proportional to the cell rate rather than the bit rate. Given that the ability to power and cool the package increasingly constrains chip design, this is a key advantage.

On the other hand, optical switching has three main drawbacks: the absence of a practical, dense, fast-access optical-memory technology; the complexity of optical control and arbitration; and the cost of fast optical-switching components. In the Osmosis project, our objective is to address these drawbacks by adopting a hybrid electro-optical approach using electronics for buffering and scheduling, and optics for transmission and switching. First, we plan to solve the technical challenges of building a demon-

strator interconnection network that meets a specific set of ambitious target requirements, using components and technologies currently available. Second, we aim to accelerate the cost reduction of all-optical switches for HPC by achieving denser optical integration and finding—in addition to the low-volume HPC market—a high-volume market for these optical components. In previous publications, we explained how we chose the switch architecture and showed that the system design is feasible.<sup>1-3</sup> Here, we address the technical issues we encountered in the electronic domain.

## Osmosis architecture

First, we review the Osmosis interconnection network's architectural requirements, our resulting design choices, and the general architecture.

### Requirements

HPC interconnects must handle the arrival, buffering, routing, forwarding, and error management of high-rate data and control streams in scalable local networks. HPC switches must deliver very low latency, an extremely low bit-error rate, a high data rate, and extreme scalability. Moreover, they must handle bursty traffic efficiently, with demands ranging from very small messages (such as collectives and syncs) to bulk data set transfers (such as memory pages).

The specific HPC-based requirements for our demonstrator system include

- less than 1- $\mu$ s latency, measured application to application;
- switch ports operating at 40 Gbps with 75 percent available for user payload;
- $10^{-21}$  bit-error rate;
- optical data path;
- scalability to at least 2,048 nodes and to higher line rates (at least 160 Gbps); and
- FPGA-only implementation of all electronic logic for flexibility and acceptable cost.

The system's memory model imposes additional requirements. Shared-memory, cache-coherent, HPC multiprocessor systems, whether they have uniform or nonuniform memory access, rely on hardware support to enforce memory coherency and cache-line

granularity consistency. Consequently, the interconnection network needs the following:

- Communication must be fine-grained; typical coherent transactions comprise a total payload of 100 to 300 bytes, excluding the interconnection network's internal overhead.
- Transaction ordering must usually be strictly enforced. Although costly to implement in hardware, the sequential consistency model is generally accepted, so supporting it in the interconnection network is a strong benefit.
- Efficient support for multicast and broadcast is a basic requirement for snoop-based multiprocessors as well as collective and synchronization operations—barriers, reductions, locks—used in HPC applications.

### Design choices

We previously provided a top-down exposition of the design decisions shaping the Osmosis architecture.<sup>2</sup> Let us briefly review the key points.

*Switch degree.* The most important requirement in an HPC interconnection network is low latency, typically measured application to application across the network. We had to design the system to achieve low end-to-end latency, even with 2,048 nodes. Scaling a single-stage network to 2,048 nodes is generally not cost-effective, because of the quadratic complexity involved, so we turned to multi-stage topologies. Given our challenging 1- $\mu$ s latency target, we chose a fat-tree topology, which has the advantages of being nonblocking and offering high path diversity and short paths to neighboring nodes. We selected the switch degree  $N=64$ , which allows scaling to 2,048 nodes with 96 switches in only two levels; thus, there are at most three hops between any two nodes.

*Switching paradigm.* The latency and efficiency requirements were the main determining factors in our choice of cell switching. In this respect, Osmosis is fundamentally different from other optical switches, most of which use techniques such as meshing, circuit switching, provisioning (bandwidth reserva-

tion), or aggregation (burst or container switching). These approaches are prohibitively expensive (meshing), are too inefficient (meshing and circuit switching), or introduce too much latency (aggregation). Instead, Osmosis switches fixed-size cells by reconfiguring the optical core on a cell-by-cell basis. The key enabler is the fast switching time (a few nanoseconds) achieved by state-of-the-art semiconductor optical amplifiers (SOAs), which now, for the first time, allow low-overhead optical cell switching.

*Cell overhead.* We selected a 256-byte fixed cell size. The cell format comprises user data and overhead, which includes SOA switching time, preamble, cell header, forward error correction (FEC), line coding, and jitter budget. The resulting efficiency is about 75 percent. The Osmosis demonstrator can sustain full-link bandwidth with traffic consisting only of 256-byte cells.

*Switch fabric structure.* SOAs are well suited to implement the optical equivalent of a crossbar using a broadcast-and-select (B&S) data path architecture combining space- and wavelength-division multiplexing.<sup>1</sup> A B&S architecture is strictly nonblocking, inherently offers broadcast capability, and can (independently) scale in the space and wavelength dimensions. The crossbar equivalence determines queuing and scheduling.

*Buffer placement.* Because optical buffering is not yet mature, we cannot satisfactorily resolve contention for such a large switch degree ( $N = 64$ ) in the optical domain. We therefore decided to use electronic buffers placed at the inputs as well as the outputs of the optical core. To eliminate head-of-line (HOL) blocking at the input buffers, we arrange them as virtual output queues (VOQs). These buffers are located in the corresponding port's adapter card, which performs all interfacing functions between the attached computing node and the interconnection network. We distinguish between ingress and egress adapters, although they are typically located on the same card.

*Maximum throughput.* Maximum throughput is a relative measure that expresses how much of the aggregate raw data rate can be used with-

out saturating (overloading) the switch. This is determined by the traffic pattern, which we cannot influence, and the control path's scheduling-related properties—the queuing structure, the control channel protocol, and the scheduling algorithm. The first step toward high throughput is to prevent HOL blocking, which we accomplish with the input queues' VOQ organization. The second step is to ensure that the scheduler has an accurate view of the current VOQ state, which we accomplish with a special control channel protocol (an incremental request grant protocol, or  $\Delta$ RGP).<sup>4</sup> The third and final step is to use centralized, efficient scheduling.

*Scheduling.* The cell-switching nature of the switch implies that for high maximum throughput and low latency, the switch needs a centralized scheduler to compute a high-quality (near-optimal) matching of inputs and outputs. This problem has been studied extensively, and there are practical heuristic, iterative algorithms to solve it.<sup>5,6</sup> This class of algorithms achieves up to 100 percent throughput under uniform traffic patterns, although they are not strictly work-conserving for any arbitrary traffic pattern. However, they need  $O[\log_2(M)]$  iterations to converge on a maximal matching. The challenge is to perform the full scheduling process once in each time slot of 51.2 ns (2,048 bits at 40 Gbps). Compounding this problem is the large switch radix, which means not only that each iteration takes longer (because more ports must be considered) but also that achieving satisfactory latency and throughput characteristics requires more iterations. To increase both the scheduling rate and quality, we use the FLPPR (fast, low-latency, parallel, pipelined arbitration) scheme, which operates a set of identical subschedulers in parallel. The Osmosis demonstrator doesn't support traffic priorities, yet.

### Single-stage architecture

Figure 1 shows the high-level Osmosis switch architecture in a single-stage configuration. It supports 64 nodes and operates in synchronous, time-slotted fashion with fixed-sized cells. Adapter cards, which form the interface between the computing nodes and the interconnection network, include an

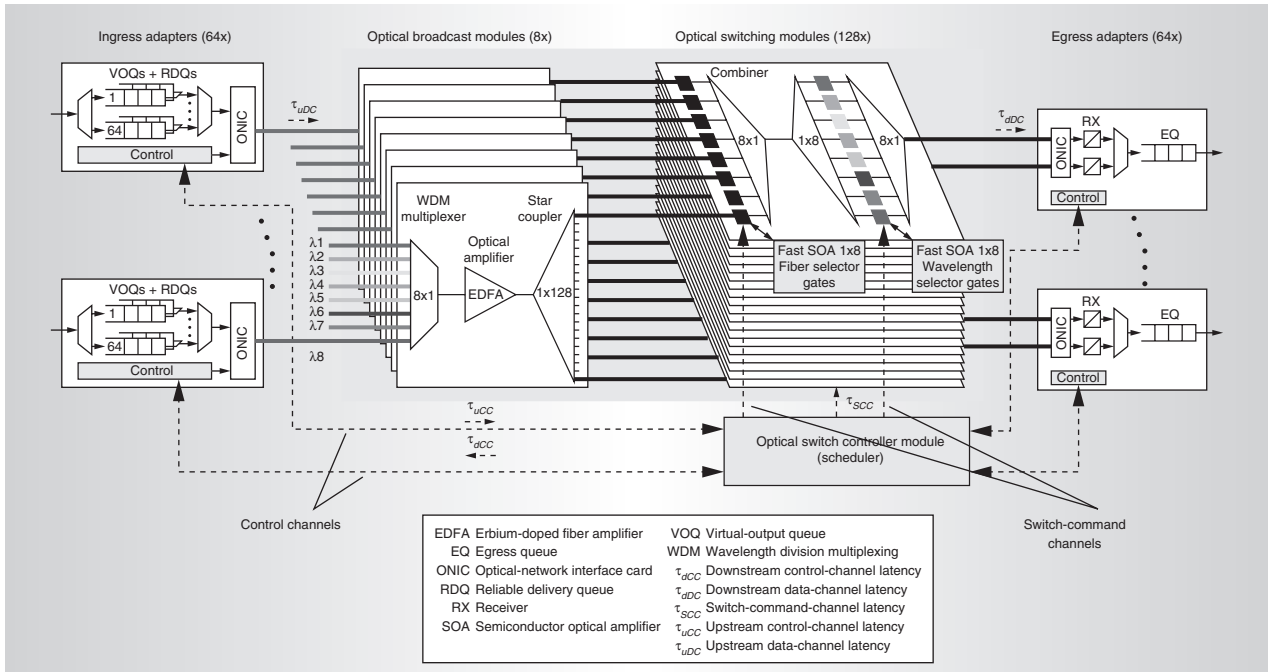


Figure 1. Osmosis architecture.

optical-network interface card (ONIC) for electro-optical and opto-electrical conversions. Each adapter consists of a full set of  $N$  VOQs with a corresponding set of  $N$  reliable delivery queues, as well as an egress queue. (Although Figure 1 shows separate ingress and egress adapters, they typically share the same card.) Each adapter has a dedicated optical control link to the centralized scheduler, which carries the control channel protocols. The scheduler resides on a separate card close to the optical switching modules (OSMs), which it configures via the switch command channels (SCCs).

We implemented the routing function with a B&S architecture that combines eight-way space-division and eight-way wavelength-division multiplexing to form a 64-port fabric. We assign each adapter a specific wavelength,  $\lambda_i$ ; we group the adapters by eight so that each adapter in a group has a unique wavelength. The optical signals per group are multiplexed onto a single fiber. An erbium-doped fiber amplifier (EDFA) amplifies the resulting signal, which a 128-way coupler then broadcasts to 128 OSMs.

Each OSM has fiber and wavelength selection stages. The fiber selection stage consists of eight SOAs, which select the fiber that carries the cell to be received. A planar light-wave

circuit combines the output of all SOAs to a common port, and the resulting stream is then demultiplexed and passed through the wavelength selection stage (also consisting of eight SOAs), which selects the desired wavelength. The stream is then remultiplexed onto a common output port and delivered to the broadband receiver residing at the egress ONIC. Synchronously with cell transmission, the centralized scheduler resolves contention on requests it receives via optical control channels and provides control signals (via the SCCs) that drive the fiber and wavelength selection SOAs just as the cells arrive at each OSM.

A special feature of Osmosis is the presence of two OSMs and two receivers per egress adapter, which allows delivery of up to two cells to the same adapter in one time slot; in other words, the output speedup equals 2. Hence, the crossbar is asymmetric; it has 64 inputs and 128 outputs. The egress adapters handle the required rate conversion. The purpose of this arrangement is to improve performance, as we will explain later.

To prevent overflow of the egress buffers, we use an on/off flow control (FC) loop between the egress buffers and the scheduler. The FC information is embedded in the upstream control-channel messages.

### Multistage support

As explained earlier, we use a fat-tree network to scale to 2,048 or more nodes. Certain provisions in Osmosis support this network. First, the cell header contains source-routing tags for up to three stages, link-level on/off FC (LL-FC), and acknowledgments (Acks) for link-level interswitch reliable delivery (RD).

Osmosis's cell-switching nature implies that contention will occur at each stage; hence, buffering is required at each stage. Despite the cost, power, and latency penalty, electronic buffers between stages are inevitable, necessitating electro-optical and opto-electrical conversions. To achieve lossless communication, the network must also be bidirectional, enabling in-band, closed-loop LL-FC for all connections between subsequent stages.

At each stage, input and output ports exist in pairs implemented in the same physical adapter card. The LL-FC for the upstream direction is embedded in the headers of the cells traveling downstream, and vice versa. The receiving ingress adapter extracts the LL-FC and passes it to its paired egress adapter. The ingress adapter also informs the egress adapter of the FC status, depending on its current buffer state, so that the egress adapter can include this information in the next outbound cell header. The FC information is derived from a programmable threshold, which should be set low enough to avoid buffer overflow, and high enough to prevent underflow.

### Control path challenges

The remainder of this article focuses on the control path, specifically on the scheduler's design and implementation. The four core requirements that drive this design are latency, throughput, multicast, and reliability. Because of the combination of ambitious targets (large switch radix, very low latency, high cell rate) and an FPGA-only implementation, we couldn't apply existing solutions in a straightforward manner. Here, we explain how we addressed these challenges.

### Latency

As latency is the key requirement in HPC applications, we first identify the latency components and explain how we have optimized latency in the Osmosis architecture.

*Latency components.* Per-hop switch latency has two components: control path latency (time spent in the input queue from arrival until departure) and data path latency (time from departure from the input queue until arrival at the local destination port). Data path latency is traffic independent; it consists of the up- and downstream data channel's physical-layer latencies ( $\tau_{uDC}$ ,  $\tau_{dDC}$ ), which include all latencies incurred in the optical switch. Control path latency consists of upstream control-channel latency ( $\tau_{uCC}$ ), scheduling latency ( $\tau_{sch}$ ), and downstream control-channel latency ( $\tau_{dCC}$ ). Upstream control-channel latency is the time from cell arrival until the scheduler is aware of this arrival. Scheduling latency is the time the request spends in the scheduler until it is granted. Downstream control-channel latency is the time from grant issuance to cell launch. Like data path latency,  $\tau_{uCC}$  and  $\tau_{dCC}$  are constant factors that depend only on the physical layer, whereas  $\tau_{sch}$  depends heavily on traffic conditions (when contention is heavy, latency increases).

Because of the packaging limitations of high-capacity, high-degree switch implementations, overhead due to the physical layer plays an important role in overall latency. The physical bulk of the switch and the power budget force distribution over multiple racks, introducing fiber delays between the racks. I/O pin count limitations force serialization and deserialization, causing additional delays on top of propagation delays. The resulting round-trip (*RT*) times in Osmosis are significant enough to affect key design aspects.

Figure 2 illustrates the per-hop switch latencies, as well as switch-command channel latency ( $\tau_{SCC}$ ). For the Osmosis demonstrator, we estimate these latencies as follows (excluding any fiber):  $\tau_{uDC} = 386$  ns,  $\tau_{dDC} = 278$  ns,  $\tau_{uCC} = 325$  ns,  $\tau_{sch} = 51.2$  ns,  $\tau_{dCC} = 184$  ns, and  $\tau_{SCC} = 418$  ns. Hence, total latency for one cell transmission across the switch equals 1224 ns, which corresponds to the sum of the full control path ( $\tau_{uCC} + \tau_{dCC}$ ), the minimum scheduling time ( $\tau_{sch}$ ), and the full data path ( $\tau_{uDC} + \tau_{dDC}$ ). In comparison, we estimate the latency of the same optical switch system implemented with ASIC components as 280 ns. Note that the zero-load latency of a two-level fat tree corresponds to three times the single-stage latencies, and that 4.93 ns must be added per meter of

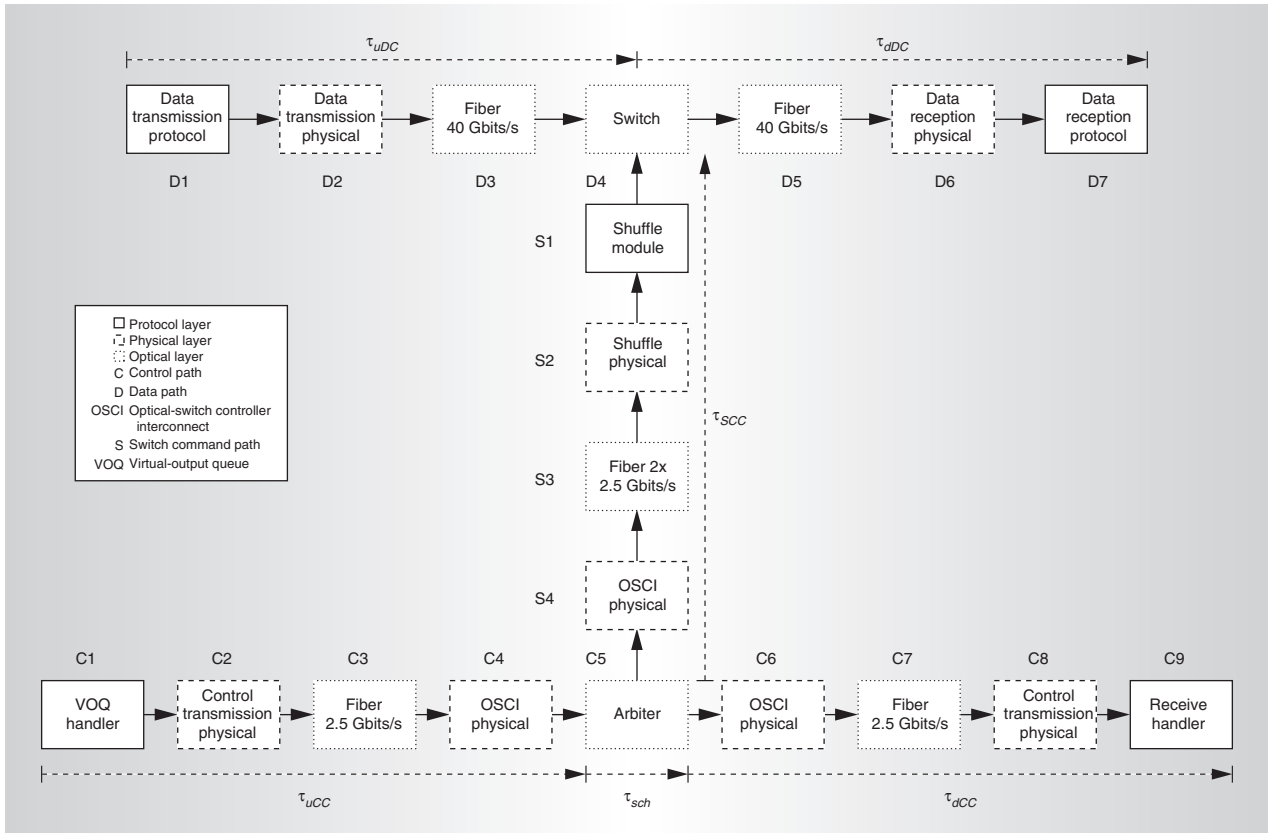


Figure 2. Latency components.

fiber. Moreover, these latencies are significant multiples of the cell duration.

The potential negative performance impact of long *RTs* can be prevented by implementing an incremental request grant protocol ( $\Delta$ RGF).<sup>4</sup> The protocol communicates relative VOQ state updates, encoded with  $\log_2 N$  bits, rather than an absolute request vector ( $N$  bits). Consequently, the scheduler must keep track of the number of pending requests per VOQ.

*Speculation.* As explained, minimum latency in Osmosis consists of control path latencies and data path latencies, which are both far greater than the cell duration. To meet the latency requirements, Osmosis features an architectural novelty called speculative transmission (STX). This scheme significantly reduces average control path latency by letting cells proceed, under certain conditions, without waiting for a grant. It operates in conjunction with a traditional centralized matching algorithm to achieve high maximum utilization, and incorporates a retransmission

mechanism to deal with failed speculations. Here, we provide a brief overview of this scheme. Precise details and an in-depth analysis are available in another publication.<sup>7</sup>

Traditionally, an ingress adapter transmits a cell only upon receipt of a scheduled grant, thus incurring full control path latency. The idea of STX is to also allow each ingress adapter to send a cell if it does not have a grant. Instead of letting the time slot go idle, the adapter selects a cell and launches it. Simultaneously, it sends a speculative request to the scheduler. To conserve the matching algorithm's desirable characteristics, the scheduler must resolve conflicts among scheduled and speculative cells so that the former are not affected. Therefore, the scheduler gives preference to scheduled connections, and only unmatched outputs can accept a speculative request. If multiple speculative cells compete for the same free output, the scheduler selects one winner—according to a round-robin policy, for example. The scheduler then acknowledges all successful speculations via the

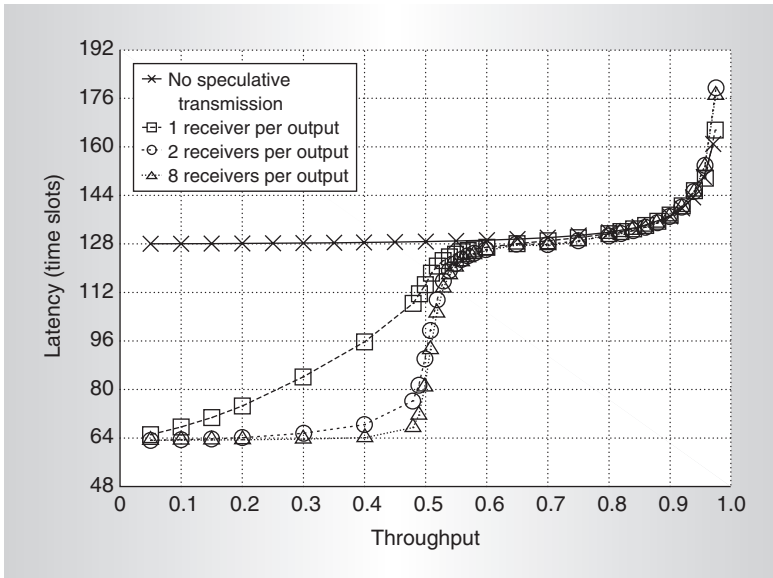


Figure 3. Delay-throughput performance using speculation with uniform independent and identically distributed (IID) Bernoulli arrivals.

downstream control channel.

To obtain the full latency benefit, we allow pipelined speculation; that is, the next speculation can start without waiting for the preceding one's result. This introduces out-of-order and duplicate deliveries. To ensure reliable, in-order, single-copy delivery, the ingress adapters have per-output retransmission queues that store the speculative cells until they are either acknowledged or granted. To avoid performance degradation and collapse due to excessive retransmissions, a cell can be sent speculatively only once. When a grant arrives, the scheduler gives cells in the retransmission queue precedence over those in the VOQ.

The second receiver that Osmosis provides at every output supports speculation: Allowing up to two cells to arrive at every output per time slot significantly increases the likelihood of a successful speculation because each output can accept one speculative cell in addition to a scheduled cell. If there is no scheduled cell, an output can accept even two speculative cells.

Our analytical and simulation results show that STX almost completely eliminates control path latency up to 50 percent utilization.<sup>7</sup> Achievable performance beyond 50 percent utilization strongly depends on the VOQ selection policy used at the input.

Figure 3 shows the performance of a generic 64-node crossbar switch employing STX, with  $RT = 64$  time slots; and one, two, and eight receivers per output. The scheduler uses an oldest-cell-first selection policy and a go-back- $N$  reliable-delivery scheme. At very low utilization, all curves with speculation exhibit 64 time slots of latency (the minimum transit latency) in contrast to 128 without speculation. As utilization increases, STX becomes less effective; but with more than one receiver, STX still reduces latency by almost half. At 50 percent load, latency increases sharply; beyond this point, delay until a speculation opportunity arises increases drastically. However, STX achieves the primary objective of reducing latency at low utilization.

### Throughput

The term *throughput* has several connotations: data rate, cell rate, and maximum utilization (saturation load). From a control perspective, the latter two present significant challenges.

*Distribution.* Achieving a high saturation load and good overall delay-throughput performance requires a centralized matching algorithm. The well-known class of parallel, iterative, round-robin matching algorithms<sup>5,6</sup> have several notable advantages: They achieve reasonable performance, they are fair, and, most important, they are generally practical. From this class, we selected the DRRM (dual round-robin matching) algorithm as a basis for the scheduler because of its proven practicality, speed, modularity, and good performance. However, a monolithic implementation for a 64-port switch is not feasible in the FPGA technology we selected.<sup>2</sup> Therefore, distribution is unavoidable. The scheduler consists of  $2N$  round-robin selectors— $N$  input and  $N$  output selectors—which are basically programmable-priority encoders.<sup>8</sup> Further sizing experiments showed that a single FPGA device can hold  $N = 64$  selectors (but not 128). This finding suggests that we can master complexity by separating the input from the output selectors.

In addition, because the input selectors operate completely independently, we can separate them from one another without consequences. Separating the output selectors from one

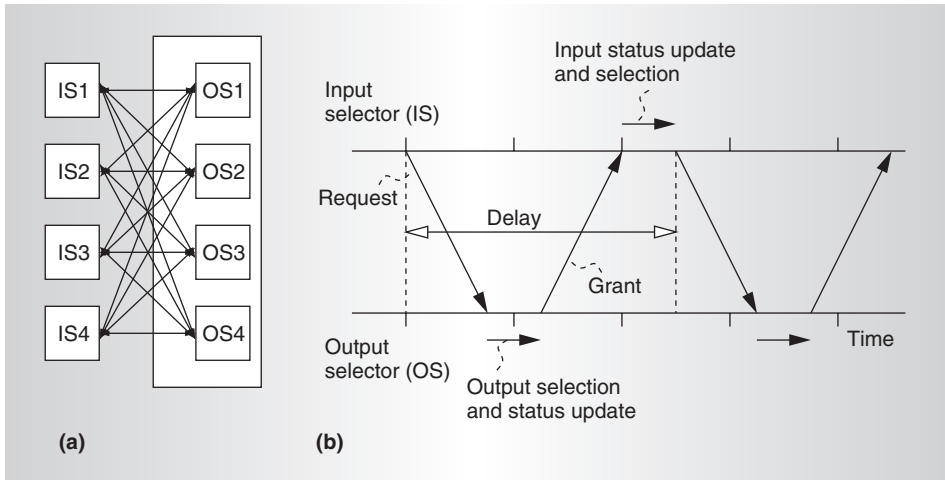


Figure 4. Distributed scheduler architecture (a); delay between input selectors and output selectors (b).

another is not feasible, because they require access to shared information (as to which inputs have been matched in previous iterations) to be effective. This leads to the distributed scheduler architecture outlined in Figure 4a.

Nevertheless, distributing input and output selectors over different devices entails some complications. In particular, it introduces a significant delay between a request and its corresponding result, as Figure 4b shows. In Osmosis, we estimate this delay at about four cell slots. By using the history of pipelined requests to select the output to request in each iteration, this distributed scheduler architecture copes with very long delays while maintaining robust performance.<sup>9</sup>

*Parallelism.* Another difficulty is accommodating the cell rate, which determines the rate at which the scheduler must produce matchings: once every 51.2 ns. Because iterative matching algorithms such as DRRM require  $O(\log_2 M)$  iterations to converge to a maximal match, the minimum cell duration  $T_c$  is a significant multiple of a single iteration's duration. When the required cell duration is too short to complete the desired number of iterations, applying pipelining or parallelism can relax this constraint. For example, two proposed schemes, pipelined maximal-sized matching (PMM)<sup>10</sup> and FLPPR,<sup>11</sup> operate  $K$  independent allocators (subschedulers) to increase the rate at which the scheduler produces matchings by a factor of  $K$ . In each time slot, one of the  $K$  allo-

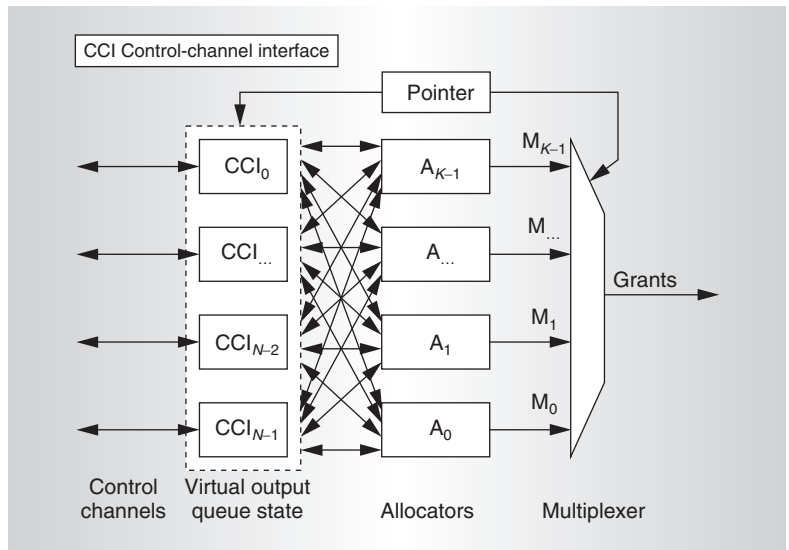


Figure 5. Fast, low-latency, parallel, pipelined arbitration (FLPPR) scheme.

cators, say  $A_b$ , completes a matching, which determines the grants to issue in the current time slot. In the next time slot,  $A_{b+1}$  completes a matching and issues it, and so on.

Figure 5 illustrates a parallel implementation of the FLPPR scheme. The scheduler contains  $N^2$  counters to keep track of the number of cells awaiting service per VOQ. We label these counters collectively as the VOQ state in Figure 5.  $K$  allocators arranged in parallel perform the matching algorithm. As a result, each allocator has  $K$  time slots at its disposal to compute a matching, thus relaxing the timing by a factor of  $K$ . We refer to a

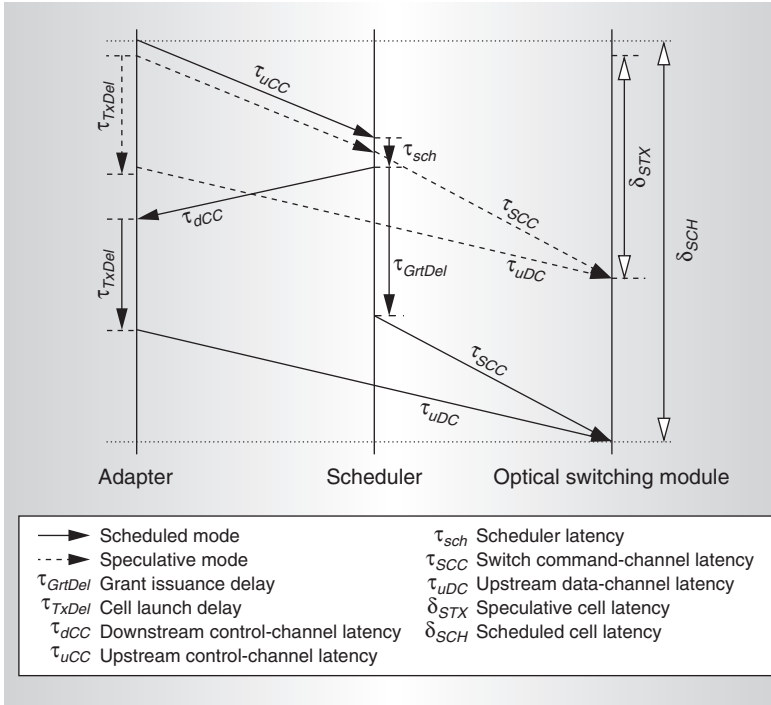


Figure 6. Grant timing.

sequence of  $K$  time slots in which a given allocator computes a matching as a matching epoch. All allocators' matching epochs are temporally staggered, such that in each time slot, exactly one allocator completes a matching epoch. At the end of an epoch, an allocator discharges its matching—which determines the set of transmission grants issued in the current time slot—and starts with an empty matching for the next epoch.

There are two main differences between PMM and FLPPR. First, FLPPR significantly reduces the mean pipelining latency penalty because it allows the VOQs to submit new requests to any stage of the (virtual) pipeline, thus contributing to meeting the latency target. Second, in PMM, the allocators don't communicate with the VOQ state unit during the epoch; no new requests are considered, and no intermediate results are divulged. In FLPPR, on the other hand, allocators can receive additional requests in each time slot, and they announce the newly added edges after each time slot. This allows immediate VOQ state updating based on new arrivals as well as scheduled departures, so that the request decisions to be made in the next slot have access to accurate, up-to-date state infor-

mation. PMM, with feedback occurring only at the end of each epoch, takes a conservative approach by assuming a priori that every request is successful. This leads to poorer performance because an unsuccessful request can be resubmitted only in the next epoch.

*System synchronization.* A large distributed system's latencies also make it difficult to ensure that the entire system is cell synchronous. We use a central reference clock, distributed to all modules via the control channel's InfiniBand physical layer. The cell launch timing must be precisely coordinated with issuance of the corresponding switch command, so that the corresponding SOAs are switched exactly in the cell gap. Figure 6 illustrates this timing issue for scheduled and speculative modes.

In scheduled mode, the following sequence of events takes place:

1. The scheduler issues a grant for a specific VOQ( $i, j$ ) via the downstream control channel to adapter  $i$ . Simultaneously, it puts the command to switch the SOAs corresponding to crosspoint ( $i, j$ ) into a delay line. The command is issued from this delay line after  $\tau_{GrtDel}$ .
2. The grant arrives at adapter  $i$ . It dequeues the HOL cell of VOQ( $i, j$ ). After delay  $\tau_{TxDel}$ , which accounts for latency incurred by grant processing, cell dequeuing, header insertion, forward error correction (FEC), and serialization/deserialization, the adapter launches the cell onto the upstream data channel.
3. After a delay of  $\tau_{scc}$  since issuance, the command becomes effective and the SOAs are switched on. Individual latencies must be precisely matched so that  $\tau_{dCC} + \tau_{TxDel} + \tau_{uDC} = \tau_{GrtDel} + \tau_{scc}$ .

In speculative mode, the event sequence is as follows:

1. The ingress adapter sends a speculative request, which arrives at the scheduler after  $\tau_{uCC}$ . The speculative cell's launch is delayed by  $\tau_{TxDel}$ .
2. The scheduler arbitrates among speculative requests for the same port, taking  $\tau_{stx}$ . Assuming there is no scheduled cell, the scheduler issues a switch command as

well as an Ack (not shown in Figure 6) for the winning request.

3. After a delay of  $\tau_{SCC}$  since issuance, the command becomes effective and the SOAs are switched on. The individual latencies must be precisely matched so that  $\tau_{uCC} + \tau_{svx} + \tau_{SCC} = \tau_{TxDel} + \tau_{uDC}$ .

## Multicast

Support for efficient multicast and broadcast communication is important for collective operations or snoop-based coherency in parallel computers. The Osmosis data path's B&S architecture is highly suitable for efficient multicast operations. However, scheduling multicast is a complicated issue. So far, there appears to be no practical algorithm that can efficiently schedule multicast and unicast simultaneously. As a result, the typical approach has been to implement separate unicast and multicast schedulers, and combine their results to arrive at a valid crossbar configuration.

Unfortunately, existing approaches either require a priori knowledge of traffic characteristics (particularly the ratio of unicast to multicast traffic) to achieve high performance<sup>12</sup> or suffer from fairness issues.<sup>13</sup> We have developed a feedback-based scheme that operates fully independent unicast and multicast schedulers in parallel. A merge unit combines the unicast and multicast matchings. Multicast has precedence over unicast, improving overall performance, because the multicast algorithm is more sensitive to restrictions in scheduling freedom. To avoid starvation of unicast traffic, unicast connections eliminated by conflicts with multicast connections (we call this set the remainder) are fed back to a filtering unit. In the next time slot, this filter removes all multicast and unicast requests that conflict with the remainder, so that in the following time slot, all unicast connections eliminated in the preceding time slot can be served. This approach attains high performance under any traffic mix, while ensuring fairness. Another publication presents further details and improvements on this basic scheme.<sup>14</sup>

The FLPPR architecture supports this approach by dedicating one of the parallel allocators to a multicast matching algorithm, such as the weight-based algorithm.<sup>15</sup> The other allocators implement unicast matching.

Moreover, the Osmosis architecture combines the merge function with the FLPPR multiplexer function in the same physical unit.

## Reliability

To meet the objective of an error rate better than  $10^{-21}$ , we adopted a two-tier approach to reliable delivery (RD). We expect the optical path's raw bit-error rate to be between  $10^{-10}$  and  $10^{-12}$ , so we use an FEC on the cell's header and data parts to reduce the bit-error rate to an estimated  $10^{-17}$ . Any errors detected but not corrected by the FEC are handled by a window-based, link-level retransmission scheme that performs up to three retries per cell. This reduces the probability of a cell's not being delivered correctly at the application level to far less than  $10^{-21}$ . Together, these schemes meet the extremely low error rate requirement.

*Data-channel coding.* We solve the problem of coding on the data channel with three elements, using a physical line code, an alignment marker, and FEC. The line code satisfies the clock-and-data-recovery unit at the receiving side by providing a defined maximum run length and a maximum DC offset in the data bitstream. The marker is a unique pattern in the data stream to lock on at a cell's start. Because the clock-and data recovery unit necessarily loses its lock at the end of each cell, the receiver finds the word alignment of the next cell's data by using this pattern. FEC corrects single-bit and detects double-bit errors in a single cell, and also detects or even corrects many higher-order errors. This approach creates about 4 percent overhead for the physical line code, 1.6 percent for the alignment marker, and 6 percent for FEC. The alignment marker can be processed on the fly, whereas the physical line code and FEC require processing latencies of about one and four cell cycles, respectively.

*Intraswitch reliable delivery.* We distinguish between intra- and interswitch RD. Interswitch RD covers the links from the egress adapters of one stage to the ingress adapters of the next stage ( $N$  connections in total). The intraswitch RD mechanism ensures that the data flow from any ingress adapter to any egress adapter within one switching node ( $N^2$  connections in total) is in order and error free.

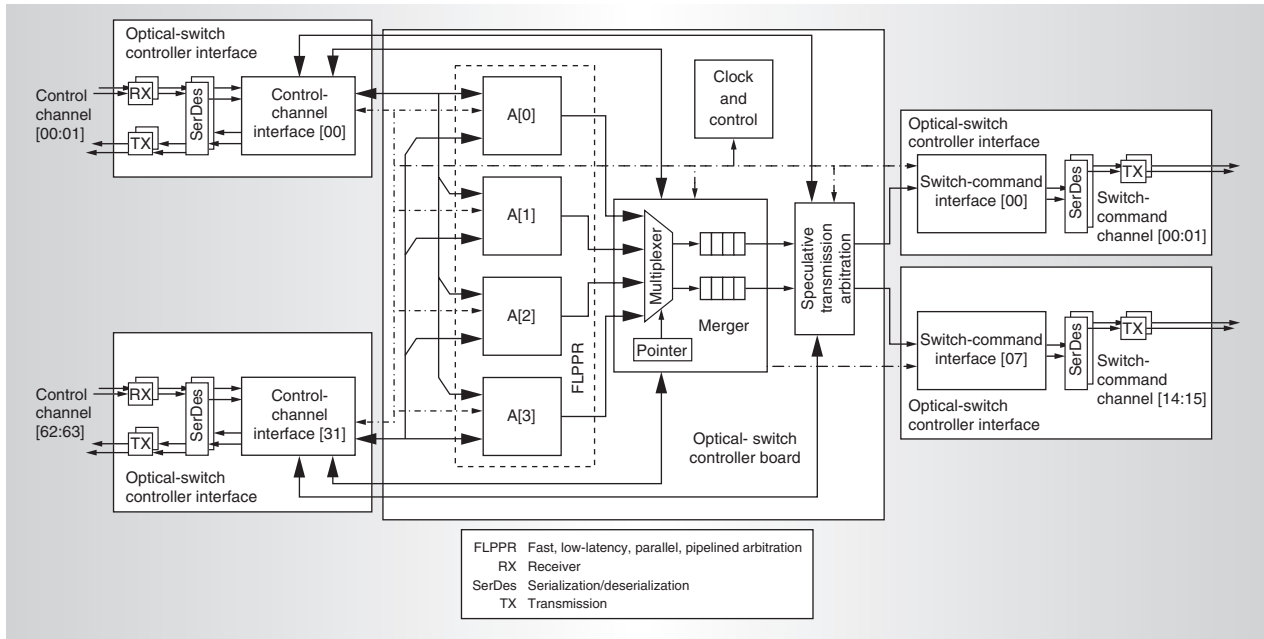


Figure 7. Logical optical-switch controller module (OSCM) architecture.

For error detection, we rely on FEC. Cells that FEC cannot correct must be retransmitted; for this purpose, the sender stores cells in an RD queue until the receiver acknowledges them. We use a go-back-N retransmission policy, with per-cell Acks being returned via the control channels. This policy has the advantage of deterministic Ack latency as well as reduced cell overhead.

*Control path reliability.* We must ensure that the  $\Delta$ RGF protocol remains consistent in the presence of errors. That is, the actual VOQ state at the adapters should be consistent with the state maintained by the scheduler, within the bounds of the  $RT$  time. To this end, we use a census mechanism that determines whether a specific VOQ's state is consistent, and, if not, indicates the error's magnitude. We also incorporated this census mechanism in the control message format.

## Implementation

The parallel and pipelined implementation of multiple unicast, multicast, and speculative schedulers presents significant challenges. In Osmosis, we refer to this combination of schedulers as the optical-switch controller module (OSCM).

In every cell slot, 96 bits of control infor-

mation must be delivered from all 64 adapter cards to the OSCM. Furthermore, grants and speculation Acks, along with census protocol updates, are delivered from the OSCM to the ingress adapters in the same cell slot. This requires an aggregate full-duplex bandwidth of 256 Gbps. The OSCM must also issue 128 switch commands to the OSMs, requiring 32 Gbps of bandwidth. The challenge is to limit the OSCM board size, while aggregating enough link transceivers to deliver the required bandwidth. We use a midplane called the optical-switch controller board (OSCB) for scheduling functions. Daughter boards called optical-switch controller interface (OSCI) cards plug into the OSCB for I/O and interface functions.

Figure 7 shows a schematic of the OSCM. It consists of 47 FPGAs, distributed as follows:

- 32 control channel interface chips, each implementing logic for two control channels (CCs) to adapter cards, including the link management unit,  $\Delta$ RGF, the census mechanism, the RD protocol, and the egress buffer FC loop. Also, CCI chips implement input-selector logic according to the distribution of the matching algorithm proposed earlier. Because of PCB area limitations, we

implement each CCI chip on an individual ( $22 \times 15 \text{ cm}^2$ ) OSCI daughter board. Each OSCI implements two physical channels, each providing a 2-Gbps, full-duplex control rate.

- *eight switch-command interface chips*, each implementing the physical-layer interface for the switch-command channel from the OSCM to an OSM. To optimize design time and cost, we reused the OSCI card and its programmable FPGA. We used eight OSCIs to transport all 128 commands to the OSMs. SCI FPGAs also implement a programmable delay line that precisely matches latencies of data, control, and switch-command channels to ensure correct and timely switching.
- *four subscheduler ( $A[0] \dots A[3]$  in Figure 7) chips*, which implement the FLPPR scheduler. Each allocator completes two iterations per time slot, for a total of eight iterations per matching epoch (or six if one of the four allocators is dedicated to scheduling multicast requests).
- *one merger chip*, which implements the multiplexing function to select grant signals from allocators, as well as the merge function to integrate unicast and multicast matchings.
- *one speculative transmission (STX) arbitration chip*, which implements STX arbitration.
- *one clocking and control chip*, responsible for system-wide distribution of a phase-synchronized reference frequency (156.25 MHz, 8 clock cycles per cell) and a cell start signal (51.2 ns), to ensure cell-synchronous operation. All synchronization is distributed from the OSCM's central high-precision clock.

Figure 8 shows the OSCM prototype's assembly with a large ( $57 \times 43 \text{ cm}^2$ ) midplane populated with four OSCI cards. The OSCB midplane has 40 slots for OSCIs, one slot for the power supply card, one slot for the micro-processor card, six scheduling FPGAs ( $A[0] \dots A[3]$ , merger, and STX), and the clocking and control FPGA. A total of 11,400 signal pins are required to interconnect the OSCB with the 40 OSCI cards. This amounts to 285 signal pins per OSCI, of which 160 signal pins (80 differential pairs) are necessary to imple-

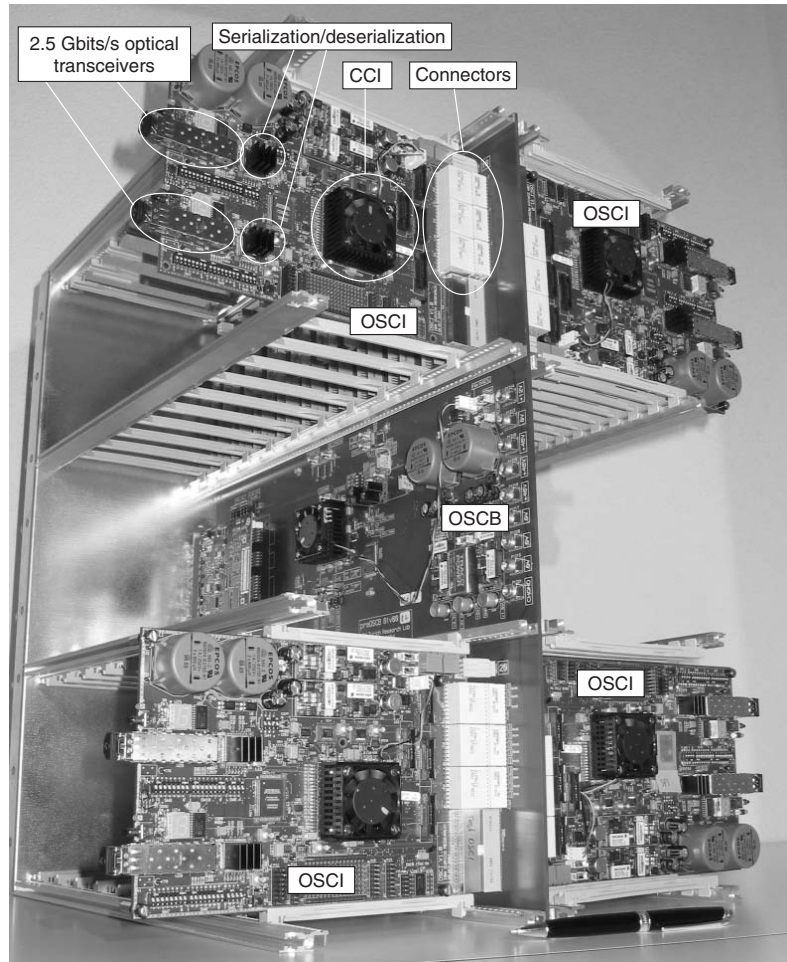


Figure 8. Physical scheduler (OSCM) demonstrator packaging, including prototype of OSCB midplane and four OSCI port interface cards.

ment the communication channels to the schedulers, 92 to implement the intraswitch RD and FC loop, and 33 for system management and control.

The number of OSCI slots and FPGAs significantly affects the OSCB's wiring and board complexity. The board contains 13,000 wires, of which 4,000 are differential pairs, for a total trace length of 2.6 km. The board requires 36 layers, of which 20 are signal and the rest are power or ground layers, and a total of 45,000 through-hole and blind vias. We used matched lengths for clocking domains. We used the Mentor Expedition autorouter to route all 13,000 nets.

The OSCM represents several architectural innovations in crossbar scheduler architecture. Although we developed these solutions

to specifically address the challenges posed by the Osmosis design, we believe they are valuable in a more general context, enabling construction of larger and faster crossbar schedulers than formerly considered feasible. The challenges remaining in the implementation of the Osmosis demonstrator are manufacturing the OSCB, integrating and testing the OSCM, and integrating and testing the OSCM with the optical data path. MICRO

### Acknowledgments

We thank our sponsors and acknowledge the technical contributions of everybody involved at IBM, Corning, Photonic Controls, and G&O. This research is supported in part by the University of California under subcontract B527064. Parts of this article were previously published.<sup>2</sup>

### References

1. R. Hemenway et al., "Optical Packet-Switched Interconnect for Supercomputer Applications," *OSA J. Optical Networks*, vol. 3, no. 12, Dec. 2004, pp. 900-913.
2. C. Minkenberg et al., "Control Path Implementation of a Low-Latency Optical HPC Switch," *Proc. 13th Symp. High Performance Interconnects (HI 13)*, IEEE Press, 2005, pp. 29-35.
3. R. Luijten et al., "Viable Opto-Electronic HPC Interconnect Fabrics," *Proc. Supercomputing (SC 05)*, IEEE Press, 2005, p. 18.
4. C. Minkenberg, "Performance of i-SLIP Scheduling with Large Roundtrip Latency," *Proc. IEEE Workshop on High-Performance Switching and Routing (HPSR 03)*, IEEE Press, 2003, pp. 49-54.
5. N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, Apr. 1999, pp. 188-201.
6. H. Chao and J. Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," *Proc. IEEE ATM Workshop*, IEEE Press, 1998, pp. 11-16.
7. I. Iliadis and C. Minkenberg, *Performance of a Speculative Transmission Scheme for Arbitration Latency Reduction*, Research Report RZ 3650, IBM Zurich Research Laboratory, 2006.
8. P. Gupta and N. McKeown, "Designing and Implementing a Fast Crossbar Scheduler," *IEEE Micro*, vol. 19, no. 1, Jan./Feb. 1999, pp. 20-28.
9. C. Minkenberg, F. Abel, and E. Schiattarella, "Distributed Crossbar Schedulers," to be published in *Proc. 2006 IEEE Workshop High Performance Switching and Routing (HPSR 06)*, IEEE Press, 2006.
10. E. Oki, R. Rojas-Cessa, and H. Chao, "PMM: A Pipelined Maximal-Sized Matching Scheduling Approach for Input-Buffered Switches," *Proc. Global Telecommunications Conf. (Globecom 01)*, vol. 1, IEEE Press, 2001, pp. 35-39.
11. C. Minkenberg, I. Iliadis, and F. Abel, "Low-Latency Pipelined Crossbar Arbitration," *Proc. Global Telecommunications Conf. (Globecom 04)*, IEEE Press, 2004, pp. 1174-1179.
12. R. Angle et al., *Multicast and Unicast Scheduling for a Network Device*, US patent 6,477,169 B1, 2002.
13. A. Andrews, S. Khanna, and K. Kumaran, "Integrated Scheduling of Unicast and Multicast Traffic in an Input-Queued Switch," to be published in *Proc. IEEE Infocom*, IEEE Press, 1999, pp. 1144-1151.
14. E. Schiattarella and C. Minkenberg, "Fair Integrated Scheduling of Unicast and Multicast Traffic in an Input-Queued Switch," to be published in *Proc. 2006 IEEE Int'l Conf. Comm. (ICC 06)*, IEEE Press, 2006.
15. B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast Scheduling for Input-Queued Switches," *IEEE J. Selected Areas in Communications*, vol. 15, no. 5, June 1997, pp. 855-866.

**Cyriel Minkenberg** is a research staff member at the IBM Zurich Research Laboratory, where he is responsible for the architecture and performance evaluation of the Osmosis crossbar scheduler. His research interests include switch architectures, networking protocols, performance modeling, and simulation. Minkenberg has a PhD in electrical engineering from Eindhoven University of Technology, the Netherlands.

**François Abel** is a research staff member at the IBM Zurich Research Laboratory, where he is responsible for the architecture and implementation of the Osmosis scheduler. His research interests include the architecture and

VLSI design of high-speed, low-latency switching systems. Abel has a BS in engineering from the École Nationale d'Ingénieurs, Belfort, France, and an MS in electrical engineering from the École Supérieure d'Ingénieurs, Marseille, France. He is a member of the IEEE.

**Peter Müller** is a research staff member at the IBM Zurich Research Laboratory. His research interests include distributed computing systems architecture, communications and interconnect technology, device physics, nanoscience, and computer modeling. Mueller has a diploma in computer science from Brugg University. He is a member of the IEEE, the ECS, and the SPS.

**Raj Krishnamurthy** is a visiting scientist at the IBM Zurich Research Laboratory, where he works on the US Department of Energy Osmosis program and the DARPA HPCS program. His research interests include operating systems, computer architecture, and system implementation complexity. Krishnamurthy has a PhD from the Georgia Institute of Technology.

**Mitchell Gusat** is a researcher at the IBM Zurich Research Laboratory. His research interests include scheduling, flow control, and congestion management for interconnection networks. Gusat has an MS in electrical engineering from the University of Timisoara, Romania, and in computer engineering from the University of Toronto. He is a member of the IEEE and the ACM.

**Peter Dill** is a researcher at the IBM Zurich Research Laboratory. His research interests include design of communications and networking system prototypes and design of high-speed and RF printed circuit boards. Dill has a diploma as a physics technician from the Swiss Federal Institute.

**Ilias Iliadis** is a research staff member at the IBM Zurich Research Laboratory. His research interests include performance evaluation of computer communication networks, traffic control and engineering for networks, switch architectures, and stochastic systems. Iliadis has a PhD in electrical engineering

from Columbia University. He is a senior member of the IEEE and a member of IFIP WG 6.3, Sigma Xi, and the Technical Chamber of Greece.

**Ronald Luijten** manages the server interconnect fabrics team at the IBM Zurich Research Laboratory, where he is working on the Osmosis optical switch demonstrator. His research interests include high-speed switching system design, including electronic crossbars and all-optical switches, and high-speed, high-density electro-optical interconnects. Luijten has an MSc in electronic engineering from the Eindhoven University of Technology.

**B. Roe Hemenway** manages Optical Physics Research at Corning Inc. His research interests include high-power fiber lasers and amplifiers, photonic crystal structures, scalable integrated WDM networks, MOEM devices silicon optical modulators, and high-speed laser diodes. Hemenway has a PhD from Stanford University. He is a member of the Optical Society of America and IEEE.

**Richard Grzybowski** is the research director of systems engineering and program management at Corning Inc. His research interests include optical interconnect networks for high-performance supercomputers, and system applications for operation in harsh environments. Grzybowski has a PhD in electrical engineering, electromagnetics and antenna design from the University of Connecticut. He is a senior member of the IEEE and president of the Finger Lakes Chapter of the International Council on Systems Engineering (INCOSE).

**Enrico Schiattarella** is a postdoctoral researcher at Politecnico di Torino, Italy. His research interests include high-performance switching architectures and interconnection networks. Schiattarella has a PhD in electronics and communication engineering.

Direct questions and comments about this article to Cyriel Minkenbergh, IBM Research GmbH, Zurich Research Laboratory, CH-8803 Rueschlikon, Switzerland; sil@zurich.ibm.com.