# Light Speed Arbitration and Flow Control for Nanophotonic Interconnects

Dana Vantrease
Univ of Wisconsin - Madison
danav@cs.wisc.edu

Nathan Binkert
HP Laboratories
binkert@hp.com

Robert Schreiber
HP Laboratories
rob.schreiber@hp.com

Mikko H. Lipasti
Univ of Wisconsin - Madison
mikko@engr.wisc.edu

## ABSTRACT

By providing high bandwidth chip-wide communication at low latency and low power, on-chip optics can improve many-core performance dramatically. Optical channels that connect many nodes and allow for single cycle cache-line transmissions will require fast, high bandwidth arbitration.

We exploit CMOS nanophotonic devices to create arbiters that meet the demands of on-chip optical interconnects. We accomplish this by exploiting a unique property of optical devices that allows arbitration to scale with latency bounded by the time of flight of light through a silicon waveguide that passes all requesters.

We explore two classes of distributed token-based arbitration, channel based and slot based, and tailor them to optics. Channel based protocols allocate an entire waveguide to one requester at a time, whereas slot based protocols allocate fixed sized slots in the waveguide. Simple optical protocols suffer from a fixed prioritization of users and can starve those with low priority; we correct this with new schemes that vary the priorities dynamically to ensure fairness. On a 64-node optical interconnect under uniform random single-cycle traffic, our fair slot protocol achieves 74% channel utilization, while our fair channel protocol achieves 45%. Ours are the first arbitration protocols that exploit optics to simultaneously achieve low latency, high utilization, and fairness.

## Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors, Interconnection architectures

## General Terms

Design, Performance

## 1. INTRODUCTION

As the number of cores on a chip increases, on-chip communication also increases. Future many-core designs must supply the bandwidth necessary to meet these demands and do so with acceptable power consumption. Optical communication channels, which span the chip and connect multiple cores, maintain high signal integrity with low power. Possible optical organizations include buses [13], circuit-switched meshes [24, 3], and a collection of single-destination, any-source channels, one per node [6, 17, 26].

### 1.1 Arbitration

When several packet sources simultaneously request the right to send on a channel, an arbiter must select one and grant access. The arbitration should provide high utilization and fair sharing with low arbitration latency, at low power and hardware costs.

Arbitration will be frequent. In a cache coherent many-core system, messages are small (cache-line size or less). Area-efficient nanophotonic waveguides and dense wavelength division multiplexing enable very wide channels, which can transmit a cache line in one or two cycles. Therefore, nearly every channel will need arbitration on nearly every cycle.

Arbitration should be fast. The time to arbitrate should not add unnecessarily or excessively to the message transfer time. In our protocols, the latency of arbitration exactly matches the latency of communication, since both use the same mechanisms.

Arbitration should be effective. It should yield high utilization of the channel. End-to-end flow control can further improve bandwidth utilization by avoiding failed transmissions due to receive buffer overflows. To address this, we devise methods to allow flow control information (buffer space reservations, or credits) to be piggybacked on the arbitration signals with little overhead.

Arbitration must be fair. Sources with equal needs should receive equal service, and none should starve. Bias in arbitration can cause longer queueing delay for packets from an unfairly treated source. We ensure fairness by dynamically adjusting the arbitration priorities when the system detects an imbalance in service.

The protocols we examine are all based on the use of an optical token that signifies temporary ownership of a single channel, *i.e.* the right to modulate certain wavelengths on certain waveguides. In its simplest form, the token is a short, single wavelength light pulse, in effect a binary "1", that travels at the speed of light in silicon (about 10 cm/nsec) through an arbitration waveguide. Interested sources attempt to read the token by removing the light, changing the "1" to a "0"; disinterested sources do not and the token propagates unperturbed. In this way our protocols take full advantage of an important property of photonic switching devices: inac-


*MICRO'09,* December 12–16, 2009, New York, NY, USA.

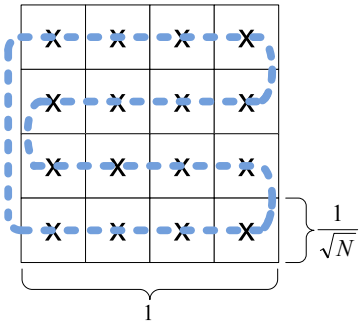



**Figure 1: Arbitration Scalability. The arbitration waveguides snake throughout chip, passing each of the $N$ cores exactly once and having length $O(\sqrt{N})$.**

tive nodes have no effect on the latency or power of messages that bypass them. Only the nodes that are participating (by actively requesting) in an arbitration are visible. Electronic arbitration protocols, as well as other optical protocols (all of which convert to electrical signals at each node), incur some latency for each node, whether interested or not. That disinterested nodes are effectively not present greatly lowers arbitration latency.

The latency of our arbiters—meaning the time between availability of a buffer at a destination and communication of this fact to a sender—is dictated by the length of the optical waveguides and the time of flight of a token through them. Since chip area is essentially fixed as core count grows with Moore's Law, each core in an $N$-core chip has area proportional to $1/N$ and has length and width proportional to $1/\sqrt{N}$. Our waveguides follow a path that visits all $N$ cores, like the one shown in Figure 1; they therefore scale in length as $N \times$ width of one core $= O(\sqrt{N})$. When $N = 64$ on a 576mm$^2$ chip with a 5GHz clock, flight time is 8 cycles. Electronic token arbiters delay the token at each node and therefore have higher latency.

The latency and power advantages of keeping the token in the optical domain at all times during arbitration come at some cost. Because optical tokens cannot be delayed, they cannot be inspected or modified as they are passing a node without removing the light from the waveguide and converting the signal to an electronic one. If the light is to be placed back on the waveguide, it must be delayed to the next clock edge. To take advantage of the fast movement of tokens in optics, they should only be removed when a node will in fact use the token. This means that nodes must use local information alone to decide whether or not a given token should be removed. So if, for example, a node is interested in two channels but can only transmit on one, it cannot claim both (by removing their respective tokens) and then release one without affecting (by delaying or consuming) the released token. Our protocols cope with these limitations.

## 1.2 An On-chip Optical Network

We apply our single-channel arbitration protocols for control of an optical interconnect that provides a dedicated, single-destination, multiple-source communication channel to each node, called a Multiple Write, Single Read (MWSR) interconnect in the Firefly terminology proposed by Pan, et al. [21]. The MWSR interconnect is a common optical interconnect

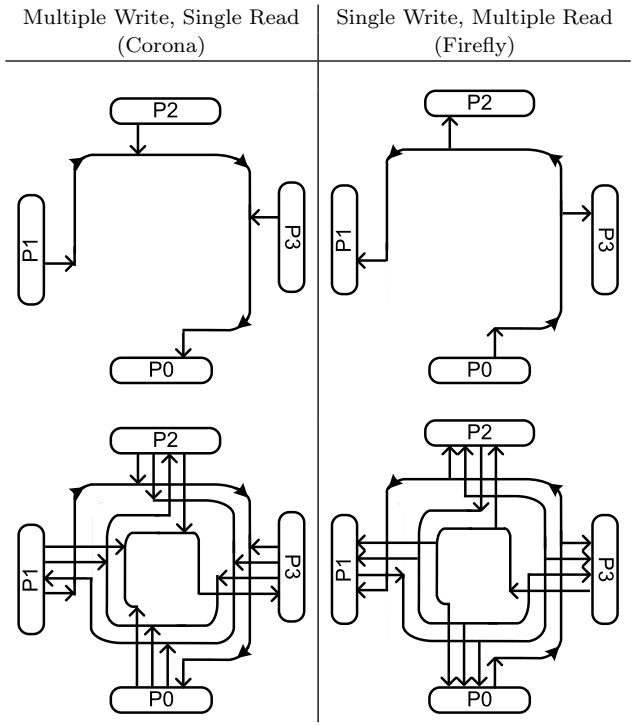



**Figure 2: Ring-based Optical Interconnects.**

architecture because it deals well with unidirectional wave-pipelined technology. Our arbiters may be applied to any optical MWSR, including [17, 26, 15, 16]. In addition, our arbiters are general enough to be applied to optical broadcast buses [13, 26]. In Section 5, we employ these protocols asynchronously on each channel of an MWSR.

In an MWSR, *any* node may write to a given channel but only *one* node (the *destination* or *home* node) may read from the channel. Optical data packets traverse the channels in a wave-pipelined, or latchless, manner from a single source to a single destination. Packets are fixed in both temporal and spatial extent. Several short packets may occupy the same channel simultaneously in moving *slots*. Time of flight varies with differing source–destination distance.

Firefly uses a Single Writer Multiple Reader (SWMR) interconnect, which is a dual to MWSR. In an SWMR, exactly *one* node may write to a given channel, but *any* node may read from the channel. Figure 2 contrasts these interconnects and shows how both provide full connectivity. An SWMR benefits from not requiring any arbitration on the part of the sender. The extra complexity, which is not present in MWSR, is that the sender must communicate to the receiver that a message is destined for it. The receiver then activates its respective detectors, which read the packet (and happen to also destroy the optical signal in the process). Firefly broadcasts a head flit to identify the designated receiver of each packet; this costs bandwidth and needs specially designed and relatively expensive broadcast waveguides and optical broadcast power.

Proposed SWMR designs may experience flow control problems, while MWSRs naturally demonstrate a degree of flow control. An MWSR node can receive at most one flit in a network cycle and as long as it can drain packets at this rate there can be no issues of flow control. On the other hand, an

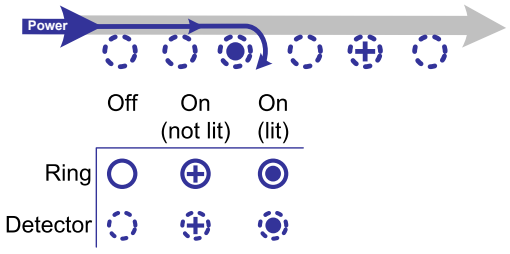



**Figure 3: Basic Optical Arbitration.**

SWMR node may receive upto $n$ flits in a network cycle—a much harder drain-rate to maintain. This paper shows how flow-control may be piggybacked onto arbitration in a MWSR.

In Section 6 we report on an experimental evaluation of the utilization, latency, and fairness of these protocols for a 64-node optical MWSR interconnect under synthetic workloads and for benchmark applications. Finally, we estimate the power used by each of our arbiters.

In summary, we make these contributions:

- We present an arbitration mechanism that for the first time satisfies all of the requirements of arbitration: it provides high channel utilization, fairly across all sources, and adds little to communication latency;
- Our mechanism takes full advantage of and copes with the limitations of optical technology described above;
- Our mechanism achieves very low latency because its tokens remain in the optical domain until consumed and used.

## 2. OPTICAL ARBITRATION

We consider optical communication structures comprised of silicon ring resonators and silicon waveguides. Waveguides confine laser light, which travels from a light source, unidirectionally, and with negligible losses. Multiple wavelengths can use the same waveguide, with no interference between them. Rings are tuned during fabrication to a particular wavelength by controlling their dimensions. When placed next to a waveguide, a ring can be used to modulate or to detect light of its particular wavelength on that waveguide, or to divert (switch) the light from one waveguide to another. The modulation, detection, and diversion functions are controlled by applying an electrical signal to the ring, which brings it into or out of resonance with its specific wavelength. We assume modulation occurs on both clock edges of the clock. Functioning ring resonators have been demonstrated [29, 30].

An activated ring detector removes all the light in the process of detecting it, thus implementing a destructive read. When the detector is inactive, the light passes the ring unperturbed. Thus, an activated detector detects a light signal if no upstream (towards the light source) detector is activated. The output of the detector is therefore a logical function of the state of all of the upstream detectors; this wired-or-like combinational operation is performed without any delay other than the time of flight of light in the waveguide.

In the simplest approach to optical arbitration, presented in Figure 3, a one-bit-wide pulse of monochromatic light travels down an arbitration waveguide. The presence of this light represents the availability of a resource: it is a token. Each node has a detector on this waveguide. Nodes that want to use the channel activate their detectors (solid- and cross- dotted rings); the other nodes do not activate theirs (empty dotted rings). At most one node can detect the token (solid dotted ring), because reading the token removes the light from the waveguide. As a result, a node detecting a token wins exclusive use of the channel. In our protocols, it uses the channel for some fixed period.

In an optical channel, modulated light, hence information, travels in one direction and arrives with slight but increasing delays at each node in sequence. To deal with this, it has been proposed [26] to send an optical clock signal through a parallel waveguide so that clock edges arrive in phase with modulated light. Nodes synch their local electronic clocks to the arriving optical clock, thereby avoiding any skew between the local clock and the optical signals. Where a signal path returns to the node at which the optical clock is generated, the data could arrive out of phase with the clock. We call this point the "dateline." By designing a waveguide whose length is an integral number of clock cycles, or by positioning a retimer at the dateline, we eliminate such a phase shift.

## 3. OPTICAL ARBITRATION PROTOCOLS

We describe our channel based, and slot based protocols which we will call Token Channel and Token Slot. We let $T$ denote the time of flight, in cycles, that it takes for an optical token to complete a full circuit from injection at its home node to detection at the same home node assuming it is not removed by an intervening node. In other words, $T$ is the number of clocks it takes the token to complete a round trip along the blue path of Figure 2.

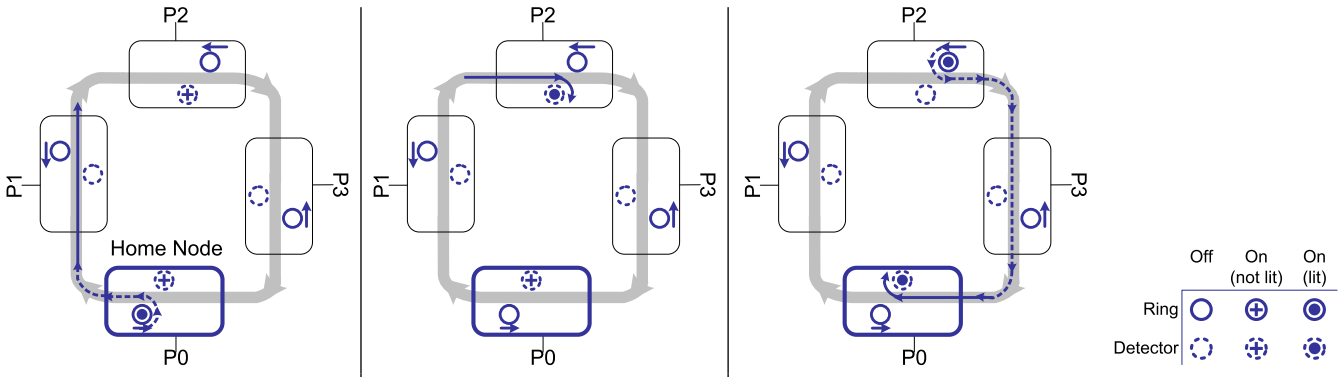
### 3.1 Token Channel

Optical Token Channel is inspired by the 802.5 Token Ring LAN standard [2]. Figure 4 shows the operation of Token Channel. There is a single token circulated per channel. The token is passed from requester to requester, entirely skipping nonrequesters that lie between. When a node removes and detects a token it has exclusive access to the corresponding channel and may begin sending (one or more) packets some number of cycles later. The sooner it begins transmitting, the better the utilization of the data channel. In Token Channel, no more than one source can use the channel at any one time: the segment of the data channel from the home node to the token holder carries no data, while the segment from the token holder back to the home node carries light modulated by the token holder. The sender may hold the token (and use the channel) for up to some fixed maximum number $H \geq 1$ of packets. When a sender has many packets in its queue for one receiver, it helps considerably to send more than one.

When a source first turns on its detector for an otherwise idle channel, it waits on average $T/2$ cycles for arrival of the token. When a single source wants to send a long sequence of packets on an otherwise idle channel, it transmits $H$ packets at a time and waits the full flight time $T$ for the reinjected token to return. The Token Slot arbiter of the next section reduces these token wait times.

### 3.2 Token Slot

Token Slot is based on a slotted-ring arbitration protocol [22]. It divides the channel into fixed-size, back-to-back slots and circulates tokens in one-to-one correspondence to slots.

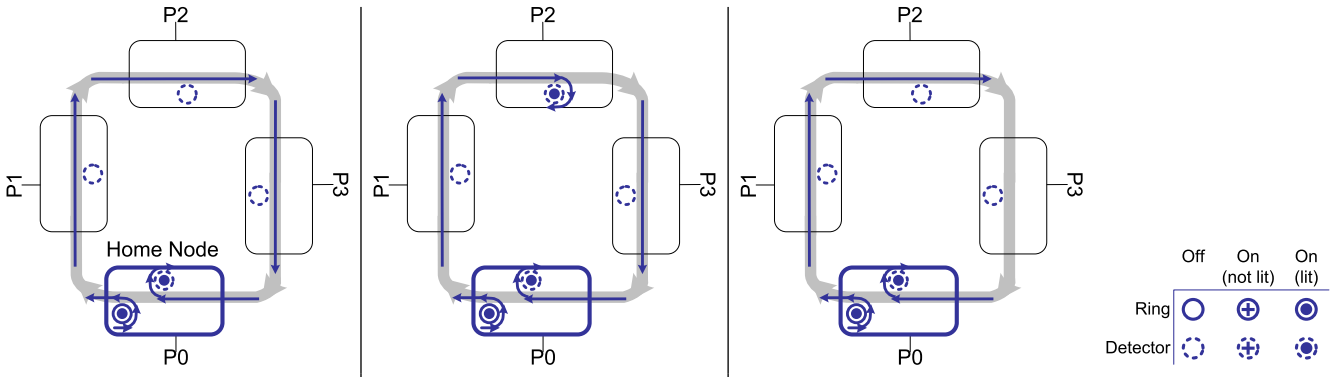



P0 injects its token onto the arbitration waveguide. It passes by P1 unperturbed. P2 requests the token.

P2 seizes the token.

P2 reinjects the token, and the token travels back to the home node (P0).

**Figure 4: Token Channel. Arbitration for one channel of an $N$ channel interconnect.**




The home node (P0) injects tokens in one-to-one correspondence for the slots.

P2 requests and seizes one of P0's token as it passes by.

The token P2 previously seized is represented by the absence of light traveling by P3.

**Figure 5: Token Slot.**

One packet occupies one slot. A source waits for arrival of a token, removes the token, and modulates the light in the corresponding single-packet slot. The token precedes the slot by some fixed number of cycles which are used to set up the data for transmission. The slot is occupied (by modulated light, *i.e.* data) until it gets to the destination, which removes the data, freeing the slot; the destination reinjects a fresh token along with new unmodulated light. Figure 5 shows the Token Slot layout and its operation.

Compared with Token Channel, Token Slot reduces the average source wait time significantly and increases channel bandwidth utilization in our experiments. In the single-source scenario above, the one active source can claim all tokens, using the channel continuously at full bandwidth. In Token Channel, the token holder is the only possible sender (there is only one token in the system), whereas in Token Slot, there are multiple tokens, and there can be multiple sources simultaneously sending at different points on the waveguide.

Unlike Token Channel, which requires each node to have reinjection capabilities on the arbitration wavelength, in Token Slot only the destination needs reinjection capability. Thus, Token Slot requires fewer resources and less power.

## 3.3 Timing

In Token Channel, after sending one or more packets, the sender reinjects a token into the arbitration waveguide in parallel with the last clock edge of its most recent transmission. If the sender cannot reinject the token at this time, so that the token lags the end of the data packets, then a data channel "bubble" results, wasting channel bandwidth. In this paper, we assume that the token can be reinjected one clock after it is removed. We intentionally chose an aggressive target to push the limits of arbitration. Early SPICE models of the analog circuitry indicate that the target is feasible.

If injecting on the trailing clock edge is too soon, a possible fix is to narrow the data channel, increasing the serialization delay of a data packet so as to cover the time needed to read, process, and reinject the token. Adding more channels could then compensate for this narrowing. Our studies indicate that halving the bandwidth and doubling the channels drops the maximum achievable throughput of Token Channel to 26% under uniform traffic, tripling and dividing by three further drops it to 18%, vs. 45% for a single channel.

We assume the node can detect a token and deactivate its detector before the next token arrives. If the ring resonator cannot respond to the token before the next token arrives, then one or more following tokens may be inadvertently claimed. Arbitration pipelining [20] can hide this effect. For example, if the ring resonators have a $k$-cycle latency, then for each data channel, $k$ arbitration channels that have tokens every $k^{th}$ slot may be employed. Since a new arbitration is started before the last one has completed, the status of the last arbitration is unknown leaving the new arbitration to speculate on whether or not it will win. Pipelining the arbitration makes the reinjection latency less critical but has a significant impact on performance. On uniform traffic, when using two- and three-cycle detectors and speculating that previous arbitrations were not won, throughput is 76% and 64% in contrast to 87% for a one-cycle arbiter.

# 4. FLOW CONTROL AND FAIRNESS

Arbitration for the interconnect is only half of the story when it comes to communicating data from source to destination. Flow control is the other half of the story: it guarantees that an outgoing packet has a receive-buffer entry allocated for its arrival at the destination. By managing flow rate, it prevents buffer overflow at the destination, avoiding the complexity and overhead of negative acknowledgements and retransmissions. In this section, we add credit-based flow control to the Token Channel and Token Slot protocols.

To be useful, a protocol must provide some guarantees of fairness. At a minimum, a fair protocol will not starve any contending user. We show here how to modify both Token Channel and Token Slot to provide fairness while retaining the advantages of fully optical implementations.

## 4.1 Flow Control

The idea is simple: the home node emits credit-filled tokens to communicate the number of available entries to source nodes. A node that removes a credit-bearing token (conveying one or more credits) is guaranteed that there are available entries at the destination.

Token Channel encodes flow control information in the token. The token is enlarged to contain a binary encoding of the number of buffer entries available at the home node. When a node detects the token, it only transmits on the data channel if there are credits available. It then marks its reservation by decrementing the number of credits in the token, and reinjects the diminished token. The token eventually returns home, at which point the home node increments the credits to reflect entries that have become available since the token's last visit. The home emits the enriched token.

Token Slot encodes flow control information simply by emitting a token only when a buffer entry is available. Thus, each token signifies a single credit and, if a node removes a token, it has also reserved an entry at the destination. When the entry is relinquished, the home emits a token.

## 4.2 Fairness

In the two protocols of the previous section, the home node emits credits. Nodes close to home have priority over nodes farther downstream in obtaining them. If emitted tokens do not have enough credits to reach distant requesters, these requesters risk starvation. This problem with token protocols is well known; it has been addressed in electronic systems with modifications that cause relatively well served senders to sit on their hands for awhile and give someone else a chance [17]. Here we modify the optical token protocols discussed above, retaining their latency advantages, and providing, by a similar back-off mechanism, fair treatment to underserved senders.

Suppose that the aggregate demand for a channel's bandwidth is less than both its inherent maximum bandwidth and the rate at which the home node frees buffer slots. Simple protocols, for example token slot, can fairly satisfy all this demand. The fairness issue surfaces when the aggregate demand by all senders exceeds this maximum achievable service rate. (In practice, full input buffers will throttle the senders so that their aggregate packet insertion rate approaches, but never exceeds, this rate.)

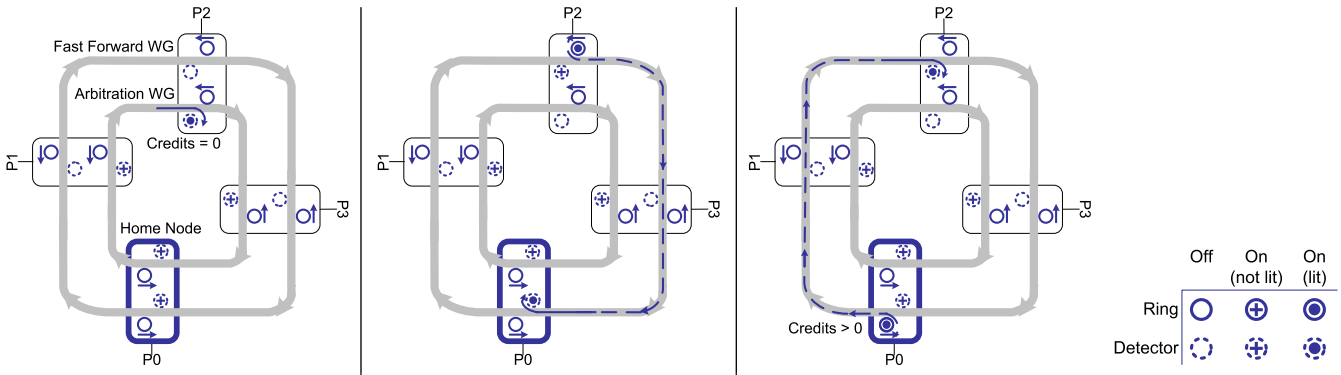
The max-min protocol [4] is a well-accepted notion of what the goal should be in this situation. In max-min, senders that need little service get all they can use. All remaining senders, which get somewhat less service than they might be able to use, get equal service.

Max-min is a worthy goal, but is difficult to achieve exactly in an online, low latency, distributed arbiter. There is no generally accepted metric for measuring the deviation between an achieved service profile and the max-min goal. We instead present the data visually, providing a clear picture of the extent to which we have approximated the max-min goal.

## 4.3 Fair Token Channel

Fair protocols such as *i*SLIP [18] implement rotating priority schemes that distribute service in some sort of cyclic manner to competing requesters. We propose to mitigate the unfair behavior of Token Channel with Fast Forward (FF) tokens that implicitly do the same sort of approximately cyclic service allocation when this is needed. When the home node injects an FF token (rather than a regular token), the token travels in its own waveguide, bypassing previously served senders. At some point along the waveguide (as described below), the FF token changes into a regular token and is available for arbitration. By fast forwarding these FF tokens downstream, nodes far from the home node are given higher priority than nodes close to home.

FF tokens are emitted onto a FF waveguide, which is concentric with the arbitration waveguide. The FF waveguide is used to both notify the home node that a node is starving *and* to supply a credit-filled token to that starving node. When a potential source node removes the token and finds it empty of credits, it declares itself to be a victim of starvation. It does not reinject the empty token on the arbitration waveguide, but rather puts it on the FF waveguide (see Figure 6). Only the home node activates its detector, and so the token quickly returns (in at most $T$ cycles) to the home node, which replenishes it with any newly available credits. Moreover, the arrival of the token at the home node via the FF channel causes the home to send the token back out again on the FF channel. The node that discovered the empty token will have activated its FF channel detector, so the replenished token will take a direct route back to that node. This not only removes the credit bias in favor of nodes near the home node, but has the added bonus of reducing the time needed to refresh an empty token with new credits. Our experiments do show a significant bandwidth improvement compared with simple Token Channel from this effect. Note that the opportunity to remove an empty token propagates cyclically, which is a key to the fairness of Channel FF.




P2 wins P0's Token from the arbitration waveguide, but does not use the channel because there happens to be no credits left.

P2 fast forwards the token to P0, past any other possible outstanding downstream requesters (*e.g.* P3). P2 also activates its detector on the FF waveguide.

When credits become available again, P0 injects its token onto the FF Waveguide. P2 is expecting the token, as shown by the active detector. It removes the token, sends, and function returns to normal.

**Figure 6: Fast Forwarding for Token Channel.**

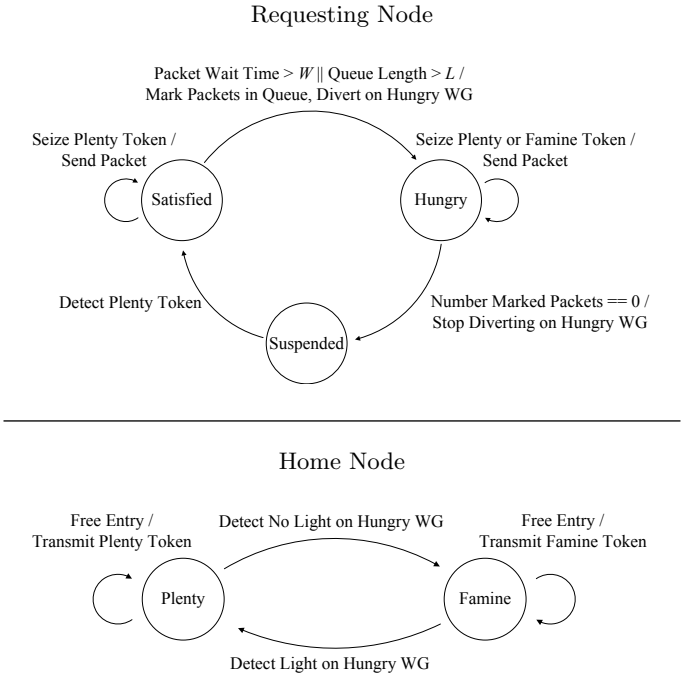



**Figure 7: Fair Slot Finite State Machine.**

## 4.4 Fair Token Slot

The simple Token Slot protocol is inherently unfair: it gives higher priority to nodes closest to the home. To ensure fairness we must make sure that every node having sufficiently high demand is treated nearly equally. Proposed electronic protocols do so using a SAT token that traverses in the direction opposite the token path and moves at most one node per clock [17]. In this section, we provide a low latency protocol, Fair Slot, that achieves this goal in a fully optical implementation. As with Channel-FF, Fair Slot detects starvation (a node that isn't receiving adequate service) and then services all underserved nodes, once each, in some sequence before returning to normal function.

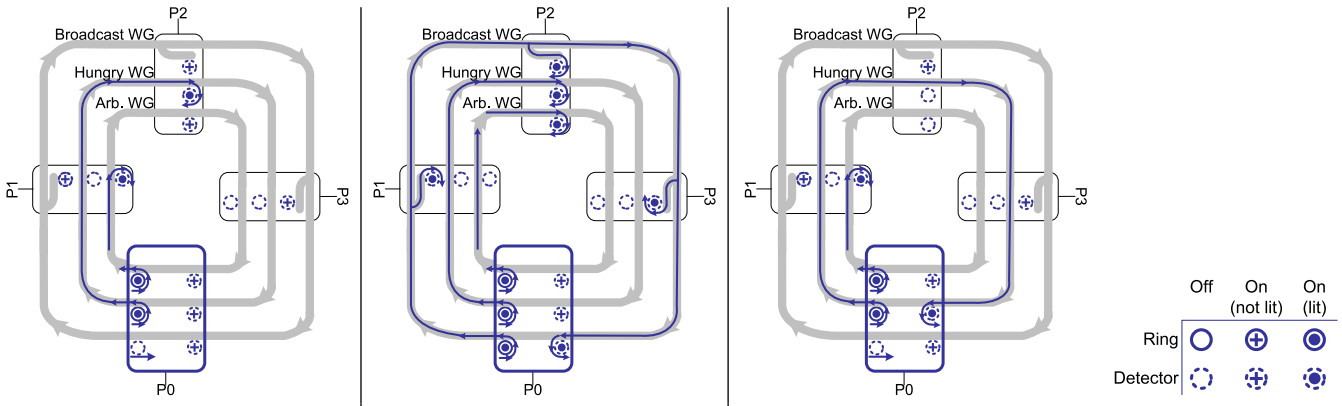
Here we describe the implementation of Fair Slot on a single channel (shown in Figure 8). In an MWSR system, each channel independently implements this algorithm.

The essential idea is to switch between simple Token Slot and an alternative that allocates credits to senders in a cyclic manner. During periods of little load, we use the simple protocol. When higher load causes some senders to receive less service than they need, the protocol switches to providing fixed quanta of credits and bandwidth to the underserved senders, once each cyclically, before returning to the simple protocol.

In Fair Slot, senders can be in one of three possible states: satisfied, hungry, and suspended. A sender must traverse the state diagram from satisfied to hungry to suspended cyclically (as shown in Figure 7). The entire channel goes through alternating phases of plenty and famine, the famine state being triggered by notification to the home node that there is at least one hungry sender. The presence of a hungry sender is communicated back to the home node using an optical *NOR* (as in Figure 3, any hungry node removes all the light from the hunger waveguide) allowing the home node to detect whether or not there is any hungry sender. When a hungry sender is detected by the home node, the channel enters the famine mode. The channel state of famine *vs.* plenty is communicated to the nodes using a separate broadcast waveguide so that all nodes can read it with no delay or reinjection.

Arrival of the famine signal changes the local (per node) channel status from plenty to famine. We call a token a famine token or a plenty token according to the accompanying channel state. Only a hungry node may grab a famine token. Other nodes (both satisfied and suspended) let them pass by. Any node can take a plenty token.

A satisfied node becomes hungry according to a local criterion. This can be the presence of an old packet (a packet is old if it has been in the system longer than some threshold) or when its input queue length exceeds some threshold. In either case, there will be an upper bound, $L$, on the number of packets in the queue of a node that becomes hungry.




The system is in Plenty mode. P1 is blocking P2 from winning P0's tokens. P2 diverts on the Hungry WG.

P0 detects no light on the Hungry WG and broadcasts on the Broadcast WG, putting the system into Famine mode. Node P1 passes on all famine tokens, allowing P2 to seize a token.

P2 becomes satisfied and releases the Hungry WG. P0 detects light on the Hungry WG, and stops broadcasting, returning the system to Plenty mode. Execution returns to normal.

**Figure 8: Fair Slot.**

When a node enters hunger for a given channel, it marks all packets in its input queue for that channel. A hungry node grabs any arriving tokens until these marked packets are flushed. Any packets that arrive after the onset of hunger can join the queue, but they aren't sent during this period of hunger. Once it sends the last of the marked packets, the node de-asserts hunger and goes into suspended state.

A node in suspended state passes on all famine tokens. When the first plenty token arrives, the node transitions to satisfied; it can also grab the token if it wishes. On the next clock it can transition to hungry if it meets the criterion.

No node can remain forever hungry. To see this, suppose node $h$ becomes hungry. It asserts its hunger and this signal must reach the home node, which transitions to famine mode if not already in it. By the current famine we shall mean the epoch of consecutive cycles, including the clock cycle at which $h$'s hunger signal arrives at home, over which the home node emits famine tokens. The famine tokens of the current famine may be taken by other hungry nodes upstream of $h$. No one of these will take more than $L$ of these tokens in the current famine, as any node that takes $L$ must have flushed its buffer (keeping newly arrived packets in the queue) and hence have gone into suspension. Node $h$ will therefore eventually receive enough packets to flush the marked packets from it input queue, suspend, and de-assert its hunger.

No node returns to hungry state before all nodes that were hungry when it became hungry leave hunger. To see this, note that the transition from suspended to satisfied requires the arrival of the broadcast that the system has transitioned to the plenty state. The home node generates this signal only when the or-ed hunger signal goes low. This cannot happen until all nodes that were hungry when the given node became hungry have suspended and de-asserted it.

When the last node goes from hungry to suspended and de-asserts hunger, there is a delay of up to $T$ cycles before home gets the word. The famine tokens generated by home will be wasted as there are no hungry senders left to take them. This has a small impact on utilization.

# 5. MWSR ARBITRATION

We evaluate the proposed optical arbiters for a $N = 64$-node, optical MWSR interconnect. The MWSR requires 64 arbiters to arbitrate for the 64 channels. In systems using dense wavelength-division multiplexing, each channel would be assigned a unique wavelength, and all could share the same arbitration waveguides. Examples of fully connected MWSR arbiters are shown in Figure 9. We consider here some engineering and scalability issues that arise when arbitrating for several channels in parallel.

## 5.1 Engineering Simultaneous Arbitration

Virtual output queues (VOQs) allow sources with requests destined for differing destinations to make these requests independently [25]. All our experiments employ VOQs.

While arbitration for one channel can occur independently of all others, no one requesting node can simultaneously transmit on all data channels because of limited per-node power. If we impose an upper bound on the number of channels that can be used simultaneously by a single sender for transmission, then how many concurrent transmissions should a sender attempt to gain in arbitration? A requester that arbitrates and wins more channels than it can use will waste unused slots and bandwidth. In Token Channel, each extra win sacrifices the immediate transmission opportunity; the requester can hold the extra tokens until it can transmit, or can reinject them; in either case this causes a channel bubble. In Token Slot, each extra token won causes the corresponding slot to go unused. Bounding the number of requests a requester may *nominate* (turn on detectors) reduces the likelihood of overwinning.

We explored the space of how many requests to nominate and how many to use with random traffic experiments. In Token Slot, being zealously conservative of bandwidth by nominating only one request and transmitting on at most one channel actually restricts the maximum achievable throughput to

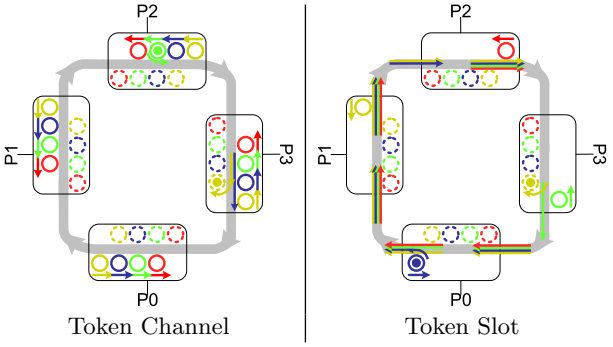



**Figure 9: Arbiters for 4-channel MWSR**

| Resource | Value |
|---|---|
| Number of nodes | 64 |
| Per node: | |
| Network Input Request Entries | 8 |
| Network Input Response Entries (SPLASH-2 only) | 8 |
| Network Output Request & Response Entries | 16 |
| Max Concurrent Nominations | 16 |
| Max Concurrent Transmissions | 2 |
| Interconnect: | |
| Packet Size (bytes) | 64 |
| Flit Rate (bytes/cycle) | 64 |
| Max Propagation Latency (cycles) | 8 |
| Bandwidth (TB/s) | 20.48 |
| Memory (SPLASH-2 experiments only): | |
| Bandwidth (TB/s) | 10.24 |
| Memory Controllers | 64 |
| Max Outstanding Requests per MC | 64 |

**Table 1: System Parameters**

58% of full channel utilization. This is due to head-of-line blocking: the nominated request prevents another outstanding request from trying for a channel, even when the channel it needs is idle. This occurs despite the presence of VOQs. Karol, Hluchyj, and Morgan [12] explain this analytically, deriving a limit of $2 - \sqrt{2} = 0.586$ on utilization. At the other extreme, nominating and using up to $N$ requests is overkill (nodes generate at most a limited number of requests per clock) and has impractical power costs. We found that nominating for all nonempty VOQs (we can have up to 8), but using at most 2 is a good power and performance point in our search space. We observe roughly 5% of tokens are wasted by this aggressive nomination strategy.

# 6. EXPERIMENTAL SETUP AND RESULTS

## 6.1 Experimental Setup

The MWSR we model operates at double data rate using a 5 GHz Clock. It has single cycle, cache-line-sized packets and optical path length $T = 8$ cycles. We assume that eight single-cycle, 64-byte non-overlapping packets can simultaneously share one channel. Table 1 shows the simulated configuration.

In order to test the proposed optical arbiters under a variety of traffic conditions, we use the SPLASH-2 workloads [27]. A 1024-thread version of SPLASH-2 is simulated: each node consists of an L2 cache and four multithreaded processors (with private L1 caches) running four threads each. The SPLASH-2 workloads model requests to and replies from memory and are trace-driven with L2 misses supplied by the COTSON simulator [9]. To ensure forward progress, responses (which are of higher priority than requests) may arbitrate regardless of the starvation-state of the system. Only requests participate in fairness throttling, since fairness of responses follows from fairness of requests For each simulation, we warm up the model and then track and measure a fixed number of requests.

We also use the synthetic Uniform and Hotspot traffic models. To isolate the performance of the arbiter, these synthetic workloads consisted of one-way traffic only and do not model memory. For these, we report the aggregate achieved packet delivery rate, the latency, and the delivery rate seen by the least serviced sender, all as a function of the offered load. The network is capable of delivering up to 64 packets, one per destination node, on every cycle. For Uniform, the destination for every packet is chosen randomly with uniform probability. An offered load of 1 means that there is, on average, for each destination, one source that generates a new request on every cycle. Values over 1 indicate that demand exceeds capacity.

For HotSpot, node 0 is the destination of all requests. Node 0's channel can sustain at most one request per cycle. Thus, an offered load of $o$ for HotSpot means that at every cycle, each node generates a request with probability $o/(63)$.

We perform our experiments with the M5 simulator framework [5] to model the interconnect, including memory and queueing at the ports.

## 6.2 Performance

We measured performance attained with five different arbiters, The first is called Baseline and is an optical channel-token protocol that delays the token by a half-cycle at each node, regardless of whether or not the node wants the channel. It is motivated by earlier work of Ha and Pinkston [10] and Kodi and Louri [14], who considered VCSEL-based technology with an electronic hop at each node. The others are our Token Channel (with $H = 1$), Token Channel w/ FF (with $H = 1$, as used in Corona[26]), Token Slot, and Fair Slot arbiters.

As shown in Figure 10(a) & (b), the Baseline proposal has low bandwidth potential and high latency solely because the tokens must be repeated at every node. The baseline only achieves 32% utilization in our Hotspot simulations, because our token can have at most 16 credits and the full round trip for a fully utilized token is at least $16 + (N - 16)/2 + 8 = 48$ cycles. Token Channel does little better on Hotspot; it saturates under a higher load but its bandwidth eventually drops off to the level of the baseline. The bandwidth drop seen in Hotspot is actually a result of credits becoming scarce and a zero-credit token being delayed by losing requesters. In fact, Token Channel behaves exactly like baseline at high loads because every node requests the token. Fast forwarding tokens, intended to improve the fairness of Token Channel, here improves performance, because the FF waveguide quickly returns zero-credit tokens to the home node. The average token round trip drops from 48 cycles to 26 cycles under heavy load. Both variants of Token Channel, however, use a single token, repeatedly delayed in flight by senders, which causes its on-board credit count to be updated infrequently and to therefore carry increasingly stale information.

The data of Figure 10 indicate that Token Slot is the best protocol available. Its implementation is relatively straightforward and it achieves nearly the best possible throughput for both test cases across the range from very light to heav-

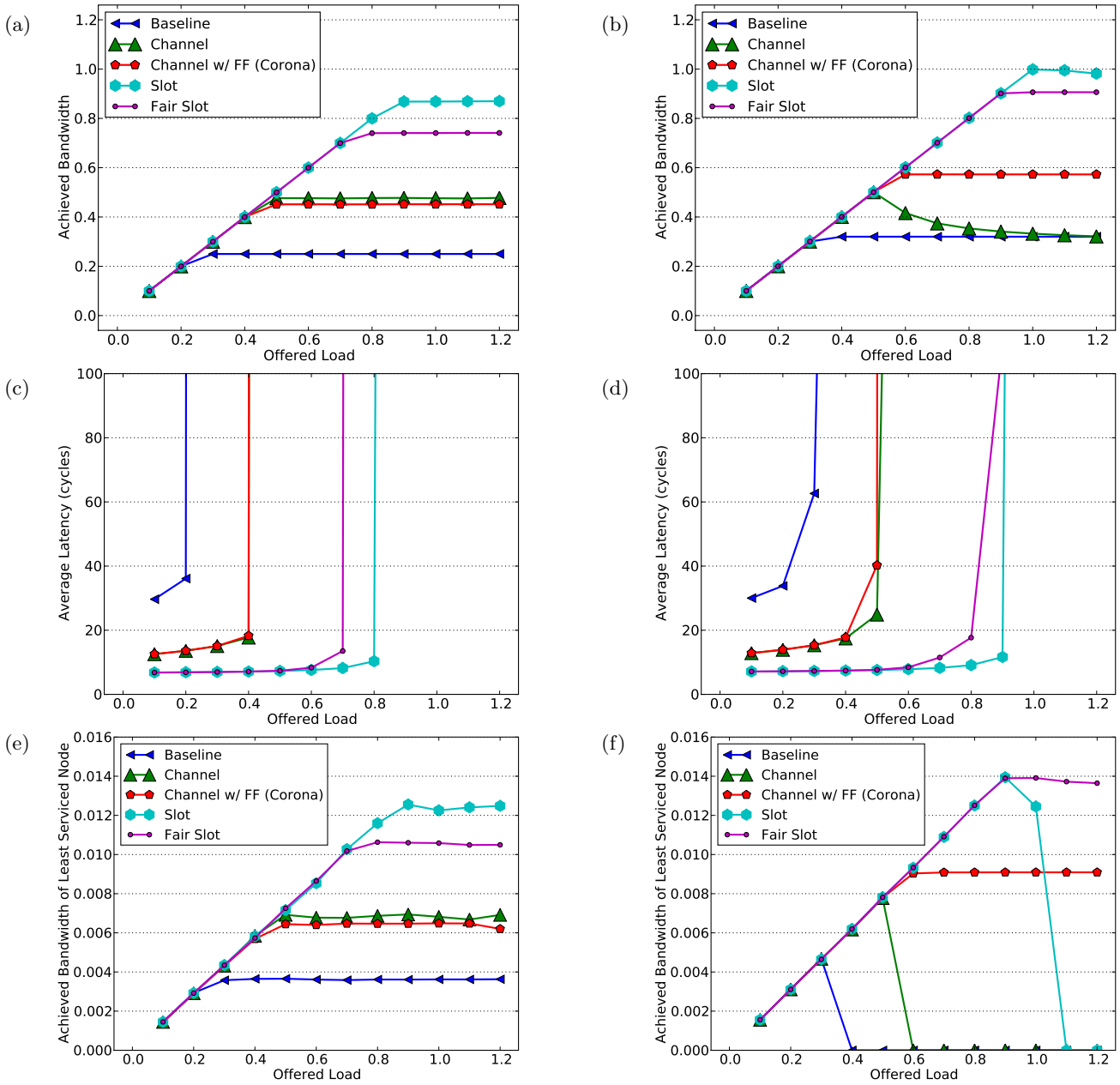



**Figure 10: Bandwidth(a & b), Average Latency(c & d), and Worst Serviced Node (e & f) for Uniform random traffic (left) and HotSpot traffic (right).**

iest achievable offered load. But Token Slot is unfair, as Figure 10(f) shows: the least-served sender starves once the aggregate demand exceeds the channel capacity. Fair Slot, which removes this unfairness, is able to achieve 90% channel utilization: 10% of available tokens (bandwidth) go unused when the system transitions modes. In HotSpot at full load, every node always has a packet available for the destination. For Uniform, Fair Slot achieves 74% utilization at maximum load: only a fraction of senders have a packet available for the destination due to small shared input queues.

Figures 10(e) and 10(f) give a closer look at how well Channel with FF and Fair Slot ensure fairness. The service given to the least-serviced sender indicates whether or not there is a starvation problem. At low loads, fairness questions do not arise – all demand is satisfied. For Uniform, all protocols remain fair even at high load. The combination of input queue size and credit availability ensure that starvation can not happen with uniform traffic. In Hotspot, at high loads, the simple protocols become unfair. Token Slot is efficient and fair below network saturation, but falls over beyond that point. Fair Slot and Channel w/ FF avoid starvation at all measured loads. They are fair: the bandwidth seen by the least served sender is close to an equal share of the aggregate bandwidths shown in Figures 10(a) and (b).

In order to confirm that Fair Slot achieves excellent arbitration latency and channel utilization with excellent fairness, we

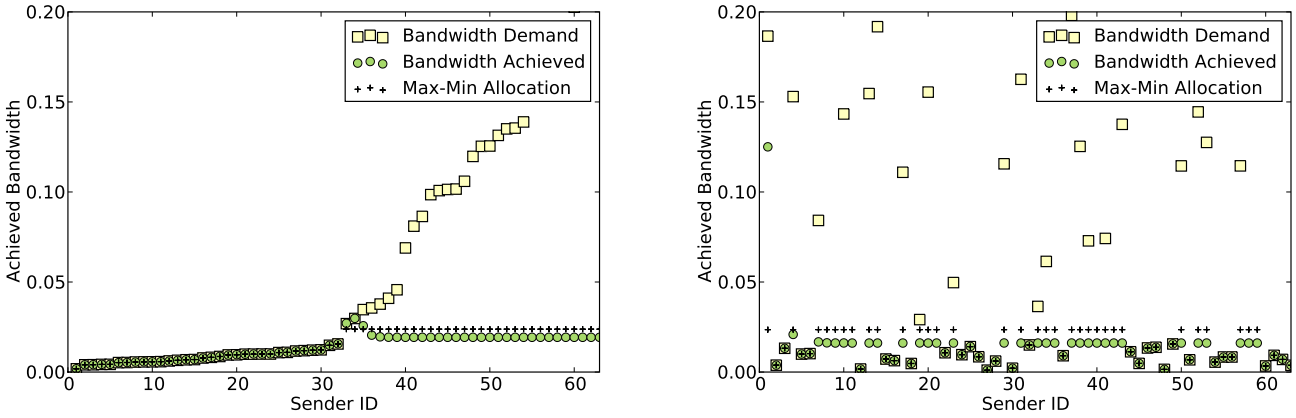



**Figure 11: Single-Channel Bandwidth Allocation by Fair Slot. Workloads use ascending demand assignment (left) and random assignment (right). Note: Excessive demands are shown at the chart's limits.**

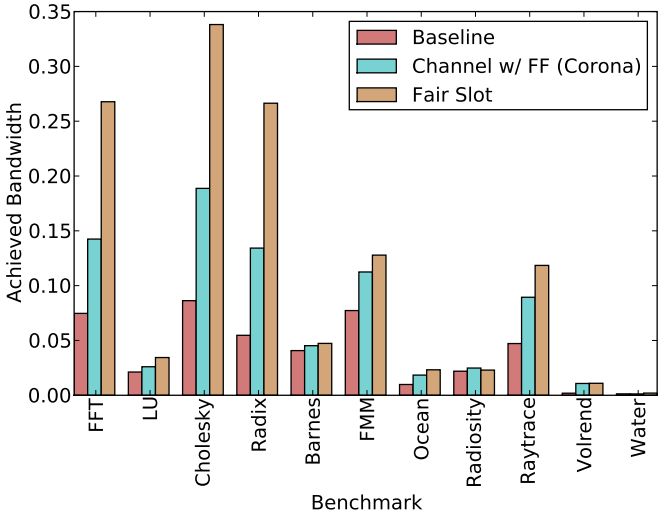



**Figure 12: SPLASH-2: Achieved Bandwidth.**

tried a more complex test case. We generated single-channel traffic in which half of senders have random, small demands, and half have random, heavy demand such that the total demand is 4 times the channel's bandwidth. Figure 11 shows achieved bandwidth: on the left, where demand is assigned to senders in increasing order, and on the right where demand is randomly assigned to senders. In both cases, the low-demand senders get all the bandwidth they need. The high demand senders are treated nearly equally, except that the two closest to the home node get more service than they deserve. But none starves, and the allocation is quite close to a max-min allocation of the aggregate achieved bandwidth. The figures also show the max-min allocation of the full channel bandwidth; the difference between the achieved service and the max-min service is caused by less than full channel utilization and the extra service for the two closest senders.

Finally, Figure 12 shows results from running 1024-threaded instances of SPLASH-2 benchmarks. Some of the workloads, such as Barnes, use the interconnect little and are indifferent to the arbitration policy.

We have presented performance results for a 64-node arbiter. Because of its scaling properties, we believe the arbiter can scale as far as an MWSR interconnect can scale. At some point, the propagation time of a transmission becomes intolerable and a hierarchical MWSR, with arbitration at each level, will be necessary.

# 7. POWER MODEL

There are two main contributors to power consumption: the laser and the rings.

Laser power comes from a static off-chip source. The laser power must overcome losses due to electrical-optical conversion inefficiencies as well as transmission losses in the waveguide. All losses target the 17nm process and were calculated with a link-loss approaching using prior results [28] and numerical simulations.

Ring resonators are the other major power budget contributor. All rings in the system must be electrically or thermally adjusted (or "trimmed") to compensate for fabrication error. Trimming helps to keep off-resonance rings in their untuned range (approximately one-half mode spacing from resonance). A ring is brought into resonance by further adjusting its index of refraction. Finally, a modulating ring will dissipate extra power, as it needs a charge of $3 \times 10^{-14}$ C/pulse at 5 GHz [1]. The ring resonators also contribute to the laser power calculation; for example, we account for the slight attenuation when a signal passes by an off-resonance ring.

Figure 13(a) demonstrates power estimates for a 64-node arbiter. Power numbers were derived using the values in Table 2. We assumed worst-case power consumption under the condition that every node was arbitrating to its full extent. We modeled two values of $m$, where $m$ corresponds to the maximum number of outstanding requests a node may nominate. Bars for $m = 16$ represent the model we arrived at in Section 5, while $m = 64$ represent an aggressive model where every node may have a nomination for each and every other node. All arbiters, except the baseline, show sensitivity to the increased activity of the ring resonators when we vary $m$. The baseline is insensitive to this change because all nodes must repeat all tokens, regardless of outstanding nominations.

Ring Trim and Ring Resonance power dominates all configurations. Ring trim stays constant while we vary $m$, because the ring count also stays constant. Allowing more rings to resonate, by increasing $m$, linearly affects the ring resonance power.

| Device | Value | Description |
|---|---|---|
| Ring Resonator: | | |
| Trim | 22 $\mu$W | Adjust for fabrication error |
| Biasing | 91 $\mu$W | "Biasing" into resonance |
| Modulation | 474 $\mu$W | Modulation at 10 Gbit/s |
| Laser: | | |
| Waveguide | | |
| Single-Mode | 1.0 dB/cm | Passing by ring banks |
| Multi-Mode | 0.3 dB/cm | Passing between ring banks |
| Non-Res. Ring | | |
| Insertion | 0.017 dB | Loss at neighbor $\lambda$ ring |
| Scattering | 0.001 dB | Loss at non-neighbor $\lambda$ ring |
| Distribution | | |
| Coupling | 1 dB | Laser coupling loss |
| Beam Splitter | 0.1 dB | Power distrib loss per-split |
| Ge RCEPD | 3.0 dB | Detector $\gamma \rightarrow e^-$ efficiency |
| Efficiency | 5.0 dB | Laser $e^- \rightarrow \gamma$ efficiency |
| Detected fJ/b | 0.15 fJ | Detecting 1k $e^-$/bit |
| Splitter Ratio | 5% | Power lost at broadcast split |

**Table 2: Optical Device Power Consumption. Note: Two-thirds of the Ring Resonator power budget is set aside for analog drivers.**

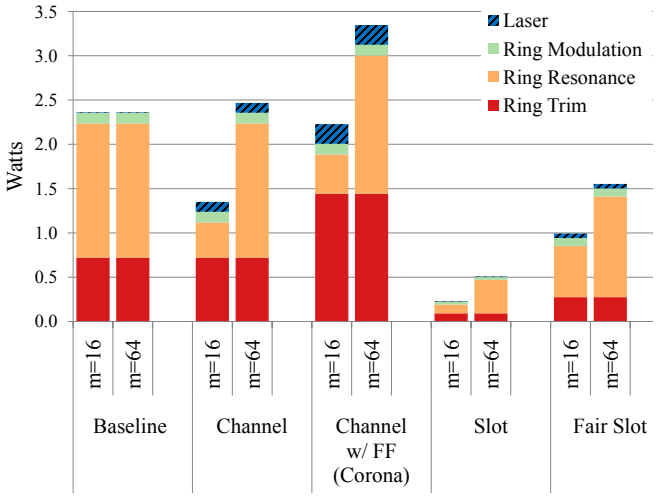



**Figure 13: Power Consumption of Arbitration Systems. $m$ corresponds to the maximum number of outstanding requests a node may nominate (or detectors it may have active).**

Configurations that have fewer rings (as inferred by the height of the Ring Trim Power bar) and that have lower values of $m$ consume the least amount of power. Fairness comes at a cost, because the protocols require more resources to be added and thus more power consumed. Nonetheless, Fair Slot consumes less than half of the power of Channel w/ FF.

## 8. RELATED WORK

Ha and Pinkston [10] and Kodi and Louri [14], advocated token-based protocols to arbitrate for optical off-chip MWSR interconnects. They processed tokens electrically at each node by converting the optical signal to an electrical signal, processing the token, and converting it back to an optical signal. Marsan et. al [17] propose a collisionless arbitration strategy for optical LAN and MAN MWSRs that relies on inspecting channel slot headers, a technique analogous to requesting tokens. The authors above targeted off-chip interconnects, which have latency and bandwidth requirements different from on-chip interconnects.

An optical arbiter, which like our proposal communicates reservations with light, can be found in Qiao and Melhem [23]. Requesting nodes send optical pulses downstream while detecting pulses that may be coming from an upstream requester. At the end of an arbitration, a successful requester detects no light (no requesters) from upstream.

Our investigation focused on arbitrating for channels individually, rather than allocating all channels in bulk. Allocation approaches to optical interconnects include Minkenberg et. al [19] and Krishnamurthy, Franklin, & Chamberlain [16]. These approaches use centralized electrical allocators that perform matchings with heuristic, iterative algorithms.

In Section 1 we discussed why SWMR interconnects do not require arbitration and at what cost. Our proposals are not directly applicable to meshes, like Shacham et. al's [24] and Cornell's Phastlane [7]. Joshi et. al [11] propose at an optical Clos network that performs all necessary arbitration in the router. Finally, Kirman et. al [13] use a broadcast-based interconnect that shares access to the medium using uses simple local state machines that are globally synchronized. We believe our arbiters may be applied to this work to dynamically adjust access according to demand.

The Metaring system [8] enforces distributed fairness in a ring topology with the use of electrical SAT(isfied) tokens. A SAT token travels in the opposite direction of data and refreshes a node's transmission quota with $k$ credits. An unsatisfied (or starved) node holds the SAT token, causing other nodes' quotas to deplete. The number of hops in a revolution has an impact on $k$, because, under normal traffic circumstances, the credits should last an entire token revolution. Although every starving node will be serviced eventually, it may take as long as $Nk$ units of time.

## 9. CONCLUSION

We have demonstrated a mechanism for optical arbitration that provides full channel utilization, fairly across all sources, and adds little to communication latency. We accomplish this by taking full advantage of optics. Our protocols leave signals in the optical domain until the information that they convey can be used. Prior optical arbitration mechanisms convert optical tokens to the electrical domain and back at each node thereby increasing the latency of the system. Leaving signals in the optical domain limits the ways in which information can be conveyed; this work has examined in detail protocols that cope with these limitations.

## 10. ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation under grant CCF-0702272, as well as grants and equipment donations from Hewlett-Packard, IBM, and Intel.

## 11. REFERENCES

[1] J. Ahn, M. Fiorentino, R. Beausoleil, N. Binkert, A. Davis, D. Fattal, N. Jouppi, M. McLaren, C. Santori, R. Schreiber, et al. Devices and architectures for photonic chip-scale integration. *Applied Physics A: Materials Science & Processing*, 95(4):989–997, 2009.

[2] ANSI/IEEE. Local Area Networks: Token Ring Access Method and Physical Layer Specifications, Std 802.5. Technical report, 1989.

[3] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popvic, H. Li, H. Smitth, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, and K. Asanovic. Building manycore processor-to-DRAM networks with monolithic silicon photonics. In *Hot Interconnects*, 2008.

[4] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[5] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The M5 Simulator: Modeling Networked Systems. *IEEE Micro*, 26(4):52–60, Jul/Aug 2006.

[6] K. Bogineni, K. M. Sivalingam, and P. W. Dowd. low-complexity multiple access protocols for wavelength-division multiplexed photonic networks. *IEEE Journal on Selected Areas in Communications*, 11:590–604, 1993.

[7] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi. Phastlane: a rapid transit optical routing network. In *ISCA '09*, pages 441–450, New York, NY, USA, 2009. ACM.

[8] I. Cidon and Y. Ofek. Metaring-a full-duplex ring with fairness and spatial reuse. *IEEE Transactions on Communications*, 41(1):110–120, 1993.

[9] A. Falcon, P. Faraboschi, and D. Ortega. Combining Simulation and Virtualization through Dynamic Sampling. In *ISPASS*, Apr 2007.

[10] J.-H. Ha and T. M. Pinkston. A new token-based channel access protocol for wavelength division multiplexed multiprocessor interconnects. *Journal of Parallel Distributed Computing*, 60(2):169–188, 2000.

[11] A. Joshi, C. Batten, Y. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic. Silicon-Photonic Clos Networks for Global On-Chip Communication. In *NOCS '09*, 2009.

[12] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *Communications, IEEE Transactions on*, 35(12):1347–1356, 1987.

[13] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi. Leveraging Optical Technology in Future Bus-based Chip Multiprocessors. In *MICRO'06*, pages 492–503, Washington, DC, USA, 2006. IEEE Computer Society.

[14] A. Kodi and A. Louri. Design of a high-speed optical interconnect for scalable shared memory multiprocessors. In *High Performance Interconnects, 2004. Proceedings. 12th Annual IEEE Symposium on*, pages 92–97, 2004.

[15] A. Kodi and A. Louri. Performance adaptive power-aware reconfigurable optical interconnects for high-performance computing (hpc) systems. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[16] P. Krishnamurthy, M. Franklin, and R. Chamberlain. Dynamic reconfiguration of an optical interconnect. In *ANSS '03*, page 89, Washington, DC, USA, 2003. IEEE Computer Society.

[17] M. A. Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri. All-optical WDM multi-rings with differentiated QoS. *IEEE Communications Magazine*, 37(2):58–66, 1999.

[18] N. McKeown. The *i*SLIP scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.*, 7(2):188–201, 1999.

[19] C. Minkenberg, F. Abel, P. Muller, R. Krishnamurthy, M. Gusat, P. Dill, I. Iliadis, R. Luijten, B. R. Hemenway, R. Grzybowski, and E. Schiattarella. Designing a crossbar scheduler for hpc applications. *IEEE Micro*, 26(3):58–71, 2006.

[20] S. S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, and D. Webb. A comparative study of arbitration algorithms for the alpha 21364 pipelined router. *SIGOPS*, 36(5):223–234, 2002.

[21] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary. Firefly: illuminating future network-on-chip with nanophotonics. In *ISCA '09*, pages 429–440, New York, NY, USA, 2009. ACM.

[22] J. Pierce. How far can data loops go? *IEEE Transactions on Communications*, 20(3):527–530, Jun 1972.

[23] C. Qiao and R. G. Melhem. Time-division optical communications in multiprocessor arrays. In *SC '91*, pages 644–653, New York, NY, USA, 1991. ACM.

[24] A. Shacham, B. G. Lee, A. Biberman, K. Bergman, and L. P. Carloni. Photonic NoC for DMA Communications in Chip Multiprocessors. In *Proceedings of Hot Interconnects*, pages 29–35, Aug 2007.

[25] Y. Tamir and G. L. Frazier. high-performance Multi-Queue Buffers for VLSI Communications Switches. *SIGARCH Computer Architecture News*, 16(2):343–354, 1988.

[26] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn. Corona: System Implications of Emerging Nanophotonic Technology. In *ISCA-35*, pages 153–164, 2008.

[27] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *ISCA*, pages 24–36, Jun 1995.

[28] Q. Xu, D. Fattal, and R. Beausoleil. Silicon microring resonators with 1.5-$\mu$m radius. *Optics Express*, 16(6):4309–4315, 2008.

[29] Q. Xu, B. Schmidt, S. Pradhan, and M. Lipson. Micrometre-scale silicon electro-optic modulator. *Nature*, 435:325, May 2005.

[30] L. Zhang, M. Song, T. Wu, L. Zou, R. G. Beausoleil, and A. E. Willner. Embedded ring resonators for microphotonic applications. *Optics Letters*, 33(7):1978–1980, 2008.