# OPTNET: A Cost-Effective Optical Network for Multiprocessors *

Enrique V. Carrera and Ricardo Bianchini

COPPE Systems Engineering
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil 21945-970

{vinicio,ricardo}@cos.ufrj.br

## Abstract

In this paper we propose the OPTNET, a novel optical network and associated coherence protocol for scalable multiprocessors. The network divides its channels into broadcast and point-to-point groups. The broadcast channels are used for memory block request, coherence, and synchronization transactions, while the point-to-point channels are utilized for memory block transfer operations. The three main distinguishing features of the OPTNET are: a) its broadcast channels behave well under high contention; b) its point-to-point channels do not require any access control mechanism; and c) it can achieve good communication performance at a low hardware cost. We use detailed execution-driven simulations of ten applications to evaluate a 16-node OPTNET-based multiprocessor. We compare our multiprocessor against highly-efficient systems based on the DMON and LambdaNet optical interconnects. Our results demonstrate that our system outperforms the DMON multiprocessors consistently for our applications, even though the OPTNET requires no more hardware than DMON. The comparison between our multiprocessor and the LambdaNet system shows performance differences that average 4% in favor of the LambdaNet. However, the LambdaNet requires a factor of $p$ more hardware than the OPTNET, where $p$ is the number of computational nodes in the multiprocessor. Based on these results, our main conclusion is that the combination of our network and coherence protocol strikes an excellent cost/performance ratio for scalable multiprocessors.

## 1 Introduction

The vast majority of parallel computers use electronic interconnection networks. However, relatively recent advances in optical technology have prompted studies on the use of optical networks in parallel computers [7, 5]. Optical fibers exhibit extremely high bandwidth and can be multiplexed to provide a large number of independent communication channels. These characteristics can be exploited to improve the performance of a multiprocessor by simply replacing its traditional, scalable network with an optical equivalent. However, optical technology can usually be exploited more effectively than this. In shared-memory multiprocessors, for instance, the broadcasting capability of optical fibers can be exploited to simplify the cache coherence hardware and protocol.

In this paper we propose the OPTNET (OPTimized OPTical NETwork), a novel optical network and associated coherence protocol that exploits all of these beneficial characteristics of optics in the design of a scalable multiprocessor. The network uses Wavelength Division Multiplexing (WDM) to provide independent high-bandwidth communication channels. These WDM channels are divided into broadcast and point-to-point groups. The broadcast channels are used for memory block request, coherence, and synchronization transactions, while the point-to-point channels are utilized for memory block transfer operations. Broadcasting memory request and coherence transactions simplifies the hardware by obviating the need for directories. In addition, broadcasting coherence transactions optimizes the coherence protocol by informing processors of changes to shared data more efficiently. Finally, our grouping of channels improves performance by decoupling the memory write traffic from the more time-critical memory block read operations.

Another optical network proposal, the DMON interconnect [7], involves broadcast and point-to-point channels that also segregate read and write operations in the multiprocessor. The two networks are also similar in terms of their number of optical components. The main differences between OPTNET and DMON-based multiprocessors are: a) our broadcast channels behave well under high contention; and b) our point-to-point channels do not require any access control mechanism and, thus, can be accessed very quickly. The LambdaNet proposal [6] can also provide both broadcast and point-to-point channels, but at a significant hardware cost: a factor of $p$ more optical hardware than our network, where $p$ is the number of computational nodes in the multiprocessor.

We use detailed execution-driven simulations of ten applications to evaluate a 16-node OPTNET-based multiprocessor. We compare our multiprocessor against systems based on the DMON and LambdaNet optical interconnects. Our results demonstrate that our system outperforms the DMON multiprocessors consistently for our applications. The comparison between our multiprocessor and the LambdaNet system shows performance differences in the range of 0 to only 12%, averaging 4%, in favor of the LambdaNet. These results

* This research was supported by CNPq/Brazil.

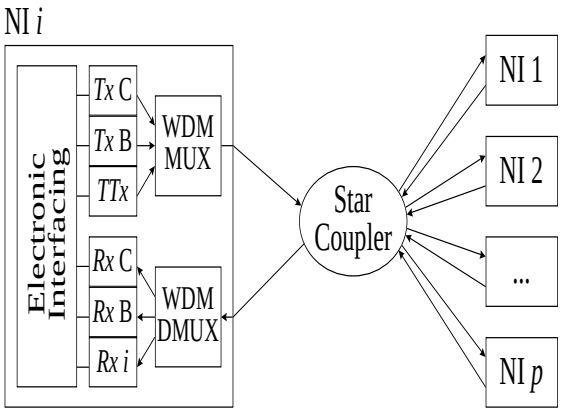



Figure 1: Overview of DMON.

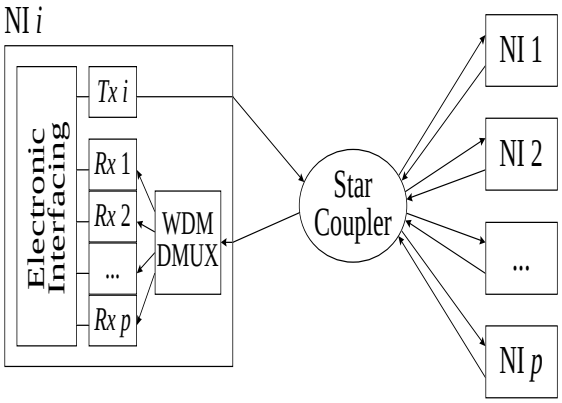



Figure 2: Overview of LambdaNet.

are exceptionally favorable to our system, given the significant difference in hardware requirements between the two multiprocessors.

Based on our results and given that optical technology will likely provide a better cost/performance ratio than electronics in the future, we conclude that the combination of our network and coherence protocol strikes a good cost/performance ratio in most scenarios.

The remainder of this paper is organized as follows. The next section presents some background material on WDM and describes the DMON and LambdaNet interconnects. Section 3 describes the architecture of our network and coherence protocol in detail. Section 4 presents our experimental methodology and application workload. Section 5 presents the results of our base experiments and parameter space study. Section 6 discusses the related work. Finally, section 7 summarizes our findings and concludes the paper.

# 2 Background

## 2.1 Wavelength Division Multiplexing

Through careful fabrication of optical fibers, transmitters, and receivers it is nowadays possible to build dispersion-free optical communication systems with low attenuation and high bandwidth. The maximum bandwidth achievable over an optic fiber is on the order of Tbits/s. However, due to the fact that the hardware associated with the end points of an optical communication system is usually of an electronic nature, transmission rates are currently limited to the Gbits/s level. In order to approach the full potential of optical communication systems, multiplexing techniques must be utilized.

WDM is one such multiplexing technique. With WDM, several independent communication channels can be implemented on the same fiber. WDM multiplexors and demultiplexors can now be found commercially with more than 100 channels. Due to the rapid development of the technology used in its implementation, WDM has become one of the most popular multiplexing techniques.

Optical networks that use WDM are called WDM networks. The simplest way of implementing a WDM network is through a star coupler and a set of receivers and transmitters. The coupler broadcasts every WDM channel to the nodes connected to the network. Nodes usually do not "listen" to all channels however, as the number of optical devices ultimately determines the cost of the network.

## 2.2 DMON

The Decoupled Multichannel Optical Network (DMON) is an interesting WDM network that has been proposed by Ha and Pinkston in [7]. The network divides its $p + 2$ (where $p$ is the number of nodes in the system) channels into two groups: one group is used for broadcasting, while the other is used for point-to-point communication between nodes. The first group is formed by two channels shared by all nodes in the system, the control channel and the broadcast channel. The other $p$ channels, called home channels, belong in the second group of channels.

The control channel is used for distributed arbitration of all other channels through a reservation scheme [3]. A node that wants to transmit on one of the channels must first wait for its turn to access the control channel and then broadcast this desire on it. The control channel itself is multiplexed using the TDMA (Time Division Multiple Access) protocol [3].

The broadcast channel is used for broadcasting global events, such as coherence and synchronization operations, while home channels are used only for memory block request and reply operations. Each node can transmit on any home channel, but can only receive from a single home channel. Each node acts as "home" (the node responsible for providing up-to-date copies of the blocks) for $1/p$ of the cache blocks. A node receives requests for its home blocks from its home channel. Block replies are sent on the requester's home channel.

Figure 1 overviews a network interface ("NI") in the DMON architecture, with its transmitters (labeled "*Tx*"), receivers ("*Rx*"), and tunable transmitters ("*TTx*"). As seen in the figure, in this architecture each node requires two fixed transmitters[1] (one for each broadcast channel), a tunable transmitter (for the home channels), and three fixed receivers (two for the broadcast channels and one for the node's home channel). The overall hardware cost of the DMON architecture in terms of optical components is then $6 \times p$.

The Snoopy Protocol Enhanced and Extended with Directory (SPEED) is a high-performance cache coherence protocol created to exploit the communication features of DMON. In its invalidate version (I-SPEED), the only version described in [7], the protocol de-

[1] These fixed transmitters are not part of the original DMON proposal. We add them here to avoid incurring the overhead of re-tuning the tunable transmitter all the time.

fines four cache and memory block states: clean, exclusive, shared, and invalid. The protocol allows only one copy of the block to be in exclusive or shared state. A node that caches a block in one of these states is the owner of the block. A cache-forwarded copy of an exclusive or shared block is received as clean by the requester. The home node of each memory block includes a directory entry that stores the current owner of the block. All misses to a memory block are sent to its home and, if necessary, forwarded to the owner node.

I-SPEED also defines states that handle critical races. A critical race is detected when a coherence operation is seen for a block that has a pending read. I-SPEED treats the race by forcing the invalidation of the would-be-incoherent copy of the block right after the pending read is completed. Further details about I-SPEED can be found in [7].

In this paper we suggest an update-based protocol for DMON. The protocol is very simple since all writes to shared data are sent to their corresponding home nodes, through coalescing write buffers. Thus, a cache miss can be satisfied immediately by the home node, obviating the need for any directory information. Our update protocol also includes support for handling critical races; it simply buffers the updates received during the pending read operation and applies them to the block right after the read is completed.

Given that a single broadcast channel would not be able to deal gracefully with the heavy update traffic involved in a large set of applications, we extended DMON with an extra broadcast channel for transferring updates. A node can transmit on only one of the coherence channels, which is determined as a function of the node's identification, but can receive from both of these channels. Besides this extra channel (and associated receivers), the hardware of the modified DMON network is the same as presented in figure 1. Thus, the overall hardware cost of this modified DMON architecture in terms of optical components is then $7 \times p$.

## 2.3 LambdaNet

The LambdaNet architecture has been proposed by Goodman *et al.* in [6]. The network allocates a WDM channel for each node; the node transmits on this channel and all other nodes have fixed receivers on it. In this organization each node uses one fixed transmitter and $p$ fixed receivers, as shown in figure 2. The overall hardware cost of the LambdaNet is then $p^2 + p$.

No arbitration is necessary for accessing transmission channels. Each node simultaneously receives all the traffic of the entire network, with a subsequent selection, by electronic circuits, of the traffic destined for the node. This scheme thus allows channels to be used for either point-to-point or broadcast communication.

Differently from DMON, the LambdaNet was not proposed with an associated coherence protocol. The LambdaNet-based multiprocessor we study in this paper uses a write-update cache coherence protocol, where write and synchronization transactions are broadcast to nodes, while the read traffic uses point-to-point communication between requesters and home nodes. Just as the update-based protocol we propose for DMON, the memory modules are kept up-to-date at all time. Again, in order to reduce the write traffic to home nodes, we assume coalescing write buffers.

Note that the LambdaNet architecture is impractical due to its hardware cost. Our only reason for including this scheme in our study is to use it as a basis for comparison against the other schemes.

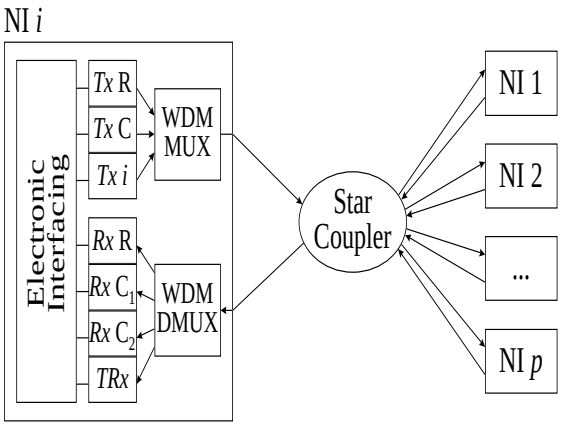



Figure 3: Overview of OPTNET Architecture.

The combination of the LambdaNet and the coherence protocol we suggest for it represents a performance upper bound for multiprocessors, since the update-based protocol avoids coherence-related misses, the LambdaNet channels do not require any medium access protocol, and the LambdaNet hardware does not require the tunning of transmitters or receivers.

# 3 OPTNET: A Cost-Effective Network

## 3.1 OPTNET Architecture

Each node in an OPTNET-based multiprocessor is extremely simple. In fact, all of the node's hardware components are pretty conventional, except for the network interface. More specifically, the node includes one processor, a coalescing write buffer, first and second-level caches, local memory, and the network interface that connects the node to the OPTNET.

Figure 3 overviews the architecture of our network. Just as DMON, our WDM network is implemented with a star coupler and divides the channels into two groups: one group for broadcast-type traffic and another one for direct point-to-point communication. Three channels, a request channel and two coherence channels, are assigned to the first group, while the other $p$ channels, called home channels, are assigned to the second group.

The request channel is used for requesting memory blocks. The response to such a request is sent by the block's home node (the node responsible for providing up-to-date copies of the block) on its corresponding home channel. The coherence channels are used for broadcasting coherence and synchronization transactions. Just like the control channel in DMON, the request channel uses TDMA for medium access control. The access to the coherence channels, on the other hand, is controlled with TDMA with variable time slots [3]. Differently from DMON, home channels do not require arbitration, since only the home node can transmit on the node's home channel.

Each node can transmit on the request channel, one of the coherence channels (determined as a function of node identification), and its home channel, but can receive from any of the broadcast or home channels. Hence, each node in the OPTNET requires three fixed transmitters (one for the request channel, one for the home channel, and the last for one of the coherence channels), three fixed receivers

(for the broadcast channels), and one tunnable receiver labeled "*TR*" (for the home channels). The hardware cost of the OPTNET is then $7 \times p$ optical components.

## 3.2 Coherence Protocol

In order to exploit the potential benefits of our network fully, the cache coherence protocol of the multiprocessor must be tailored to the network. Thus, the protocol we propose is based on update coherence, supported by both broadcasting and point-to-point communication. The update traffic flows through the coherence channels, while the data blocks are sent across the home channels. The request channel carries all the memory read requests. The description that follows details the coherence protocol in terms of the actions taken on read and write accesses.

**Reads.** On a read access, the memory hierarchy is traversed from top to bottom, so that the required word can be found as quickly as possible. A miss in the second-level cache is handled differently depending on the type of data read. In case the requested block is private or maps to the local memory, the read access is treated by the local memory, which returns the block to the processor.

If the block is shared and maps to another home node, the request is sent to the corresponding node through the request channel and the tunable receiver is tuned to the home node's home channel. When the request arrives at the home node, the home reads the block and returns it via the home channel. The requesting node waits for the block to be received, gets it from the network interface, and returns it to the second-level cache.

**Writes.** Our multiprocessor architecture implements the release consistency memory model [4]. Consecutive writes to the same cache block are coalesced in the write buffer. Coalesced writes to a private block are sent directly to the local memory through the first and second-level caches. Coalesced writes to a shared block are always sent to one of the coherence channels in the form of an update, again through the local caches.

Each update must be acknowledged by the corresponding block's home node before another update by the same node can be issued, just so the memory modules do not require excessively long input queues (i.e. update acks are used simply as a flow control measure). The other nodes that cache the block simply update their local caches accordingly upon receiving the update. When the home node sees the update, the home inserts it into the memory's FIFO queue, and sends an ack through the request channel. The update acks usually do not overload the request channel, since an ack is a short message that fits into a single request channel slot.

Finally, our coherence protocol treats the races that might result from decoupling read and write transactions by buffering updates and later combining them with the block received from memory.

# 4 Methodology and Workload

**Multiprocessor Simulation.** We simulate multiprocessors based on the OPTNET, DMON and LambdaNet interconnects. We use a detailed execution-driven simulator (based on the MINT front-end [9]) of 16-node multiprocessors. Each node of the simulated machines contains a single 200-MHz processor, a 16-entry write buffer, a 4-Kbyte direct-mapped 1st-level data cache with 32-byte cache blocks, a 16-Kbyte direct-mapped 2nd-level data cache with 64-byte cache blocks, local memory, and a network interface. (Note that the cache sizes we simulate were purposedly kept small, as simulation time limitations prevent us from using real life input sizes.)

| Operation | Latency (in processor cycles) | | |
|---|---|---|---|
| | OPTNET | Lambda | DMON |
| 1. 1st-level tag check | 1 | 1 | 1 |
| 2. 2nd-level tag check | 4 | 4 | 4 |
| 3. Avg. TDMA delay | 16 | – | 16 |
| 4. Reservation | – | – | 2* |
| 5. Tuning delay | – | – | 4 |
| 6. Memory request | 2* | 2* | 3 |
| 7. Flight | 1 | 1 | 1 |
| 8. Memory read | 44+ | 44+ | 44+ |
| 9. Avg. TDMA delay | – | – | 16 |
| 10. Reservation | – | – | 2* |
| 11. Block transfer | 22 | 22* | 23 |
| 12. Flight | 1 | 1 | 1 |
| 13. NI to 2nd-level | 16 | 16 | 16 |
| Total 2nd-level miss | 107 | 91 | 133 |

Table 1: 2nd-Level Read Miss Latency (in 5-ns pcycles) for OPTNET, LambdaNet, and DMON.

Shared data are interleaved across the memories at the block level. All instructions and first-level cache read hits are assumed to take 1 processor cycle (pcycle). First-level read misses stall the processor until the read request is satisfied. A second-level read hit takes 12 pcycles to complete. Writes go into the write buffer and take 1 pcycle, unless the write buffer is full, in which case the processor stalls until an entry becomes free. Reads are allowed to bypass writes that are queued in the write buffers. A memory module can provide the first two words requested 12 pcycles after the request is issued. Other words are delivered at a rate of 2 words per 4 pcycles. Memory and network contention are fully modeled.

In the update-based coherence protocols we simulate only the secondary cache is updated when an update arrives at a node; the copy of the block in the first-level cache is invalidated. In addition, in order to reduce the write traffic, our multiprocessors use coalescing write buffers for all protocol implementations. A coalesced update only carries the words that were actually modified in each block. All the protocol implementations assume a release-consistent model.

The optical transmission rate we simulate is 5 Gbits/s, which leads to the 2nd-level cache read miss and coherence transaction latencies listed in tables 1 and 2, respectively. Table 2 lists the latencies of the NetCache, LambdaNet, DMON with update-based coherence (DMON-U), and DMON with I-SPEED (DMON-I) systems and assume 8 words written in the cache block. All numbers in the tables are in pcycles and assume channel and memory contention-free scenarios. The values marked with "*" and "+" are the ones that may be increased by network and memory contention/serialization, respectively. The total 2nd-level cache read miss latencies in table 1 show that the LambdaNet entails 18% less overhead than OPTNET in these operations, at least in the absence of any type of contention. Under the same conditions, the OPTNET involves 24% less overhead than the DMON network in 2nd-level read misses. The total coherence transaction latencies in table 2 show that the LambdaNet entails 46% less overhead than OPTNET and DMON-I in these op-

| Operation | Latency (in processor cycles) | | | |
|---|---|---|---|---|
| | OPTNET | Lambda | DMON-U | DMON-I |
| 1. 2nd-level tag check | 4 | 4 | 4 | 4 |
| 2. Write to NI | 10 | 10 | 10 | 2 |
| 3. Avg. TDMA delay | 8* | – | 16 | 16 |
| 4. Reservation | – | – | 2* | 2* |
| 5. Update/Invalidate | 15 | 13 | 14 | 3 |
| 6. Flight | 1 | 1 | 1 | 1 |
| 7. Avg. TDMA delay | 16 | – | 16 | 16 |
| 8. Reservation | – | – | 2* | 2* |
| 9. Ack | 2* | 2* | 2 | 2 |
| 10. Flight | 1 | 1 | 1 | 1 |
| 11. Write | – | – | – | 8 |
| Total coherence transaction | 57 | 31 | 68 | 57 |

Table 2: Latency (in 5-ns pcycles) of Coherence Transactions for OPTNET, LambdaNet, DMON-U, and DMON-I. Assuming 8 words written in block.

| Program | Description | Input Size |
|---|---|---|
| CG | Conjugate Gradient kernel | 1400 $\times$ 1400 doubles, 78148 non-zeros |
| Em3d | Electromagnetic wave propagation | 8 K nodes, 5% remote, 10 iterations |
| Gauss | Unblocked Gaussian Elimination | 256 $\times$ 256 floats |
| Mg | 3D Poisson solver using multigrid techniques | 24 $\times$ 24 $\times$ 64 floats, 6 iterations |
| Ocean | Large-scale ocean movement simulation | 66 $\times$ 66 grid |
| Radix | Integer Radix sort | 512 K keys, radix 1024 |
| Raytrace | Parallel ray tracer | teapot |
| SOR | Successive Over-Relaxation | 256 $\times$ 256 floats, 100 iterations |
| Water | Simulation of water molecules, spatial alloc. | 512 molecules, 4 timesteps |
| WF | Warshall-Floyd shortest paths algorithm | 384 vertices, i,j connected w/ 50% chance |

Table 3: Application Description and Main Input Parameters.

erations, at least in the absence of contention and assuming 8 words written per block. Under the same conditions, the OPTNET and DMON-I systems involve 19% less overhead than DMON-U in coherence transactions.

Note that in our base simulations the minimum TDMA slot duration is 2 pcycles for both DMON and OPTNET networks. Thus, each control channel slot in DMON and request channel slot in OPTNET are 2 pcycles long. Each coherence channel slot in the OPTNET is at least 2 pcycles long; the actual duration of each slot depends on the number of words updated.

**Workload.** Our application workload consists of ten parallel programs: CG and Mg from the NAS suite [1], Em3d from UC Berkeley, Ocean, Radix, Raytrace, and Water from SPLASH-2 [10], and Gauss, SOR, and WF from the University of Rochester. Table 3 lists the applications and their input parameters.

# 5 Experimental Results

## 5.1 Overall Performance

Figure 4 shows the speedup of our applications running on a 16-node OPTNET-based multiprocessor. The figure demonstrates that, except for CG and WF, our applications exhibit reasonably good speedup levels on 16 nodes. Em3d, SOR, and Water, in particular, achieve excellent speedup. The two extremes in speedup performance, Em3d and WF, deserve further discussion. Em3d achieves superlinear speedup as a result of its terrible single-node 1st and 2nd-level cache behaviors; caches are simply not effective for this application on a single node. WF achieves poor performance on 16 nodes as a result of large barrier overheads due mostly to significant load imbalance.

Figure 5 shows the running times of our applications again on a 16-node multiprocessor. For each application we show, from left to right, the OPTNET, LambdaNet, DMON-U, and DMON-I performances, normalized to the OPTNET results. This figure demonstrates that DMON-U performs at least as well as DMON-I for all applications, except Water. The performance differences between these two systems average 11%, being most significant for Em3d (16%), Gauss (16%), Ocean (43%), and Radix (14%).

As one would expect, a comparison between the LambdaNet and DMON-U systems is always favorable to the former multiprocessor. Overall, the performance advantage of the LambdaNet averages 19% for our applications. SOR and Water exhibit only a small performance advantage of the LambdaNet system. For the other applications, the differences range from 16% for Gauss to 28% for CG and average 22%.

A comparison between the performance of the OPTNET and DMON-U systems is clearly favorable to our system in all cases, except SOR and Water for which the two systems perform similarly.

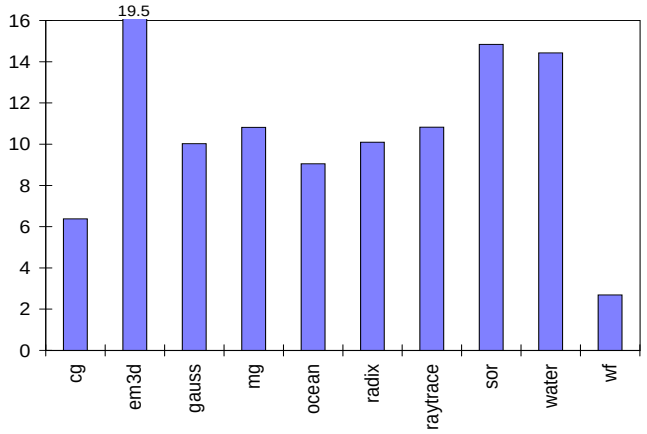



Figure 4: Speedups of 16-Node OPTNET-Based Multiprocessor.

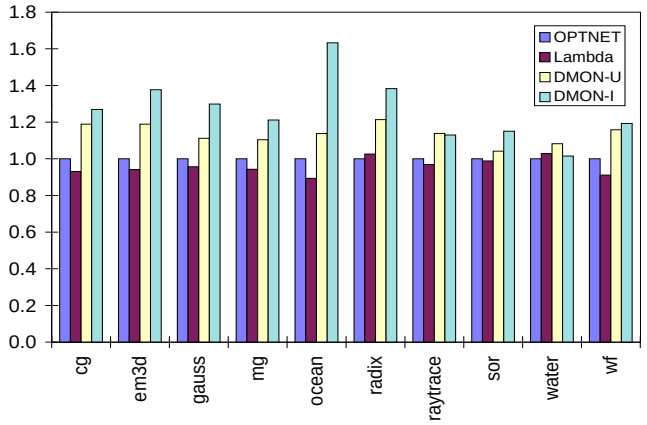



Figure 5: Run Times of (from left to right) OPTNET, LambdaNet, DMON-U, and DMON-I.

For the other 8 applications, the performance advantage of the OPTNET multiprocessor ranges from 10% for Mg to 21% for Radix, averaging 16%. Taking all applications into account, the advantage of the OPTNET averages 14%.

Figure 5 demonstrates that the OPTNET and LambdaNet multiprocessors are essentially equivalent for 4 applications: Radix, Raytrace, SOR, and Water. For the other 6 applications, the performance advantage of the LambdaNet multiprocessor is never greater than 12% and averages 8%. Taking all applications into account, the advantage of the LambdaNet averages only 4%. Given that the LambdaNet requires $\mathcal{O}(p^2)$ optical hardware, a factor of $p$ more hardware than the OPTNET, we regard these as excellent results in favor of our system.

The explanation for the performance differences presented above is the average cost of reads and writes in the various systems for each application. Thus, in the next two subsections we study these operations for all systems and applications.

## 5.2 Performance of Reads

Figures 6 and 7 concentrate our statistics on the performance of read operations on each of our systems. Figure 6 presents the average latency of read operations, while figure 7 presents the average latency of 2nd-level cache read misses. All latencies are in processor cycles. Figure 7 breaks down the average 2nd-level read miss latencies into a contention-free component ("base") and delays caused by network contention ("network"), memory contention ("memory"), and contention for off-processor-chip access and memory bus access ("other"). In both figures, the bars correspond to OPTNET, LambdaNet, DMON-U, and DMON-I, from left to right.

Figure 6 shows that the average read latency entailed by the three update-based systems (OPTNET, LambdaNet, and DMON-U) is lower than that of DMON-I, except in the cases of Radix and Water. This result can be explained in part by the fact that the update-based systems exhibit lower 2nd-level cache read miss rates than DMON-I. The differences in miss rates are not terribly significant however, since our applications are dominated by replacement misses. As shown in figure 7, the most important factor in this comparison is that read misses take longer to satisfy in DMON-based systems than in the OPTNET and LambdaNet systems, even in the absence of contention. Furthermore, the DMON-I multiprocessor suffers more significantly from memory and network contention than the other systems. For instance, discarding the Radix and Water results, DMON-I exhibits overall 2nd-level read miss latencies that are longer than the OPTNET latencies by 42% on average, while their contention-free latencies only differ by 24%. Network and memory contention are more pronounced in the DMON-I system due to writebacks of dirty cache blocks, the directory lookups required in all memory requests, and the extra messages involved in forwarding requests to the current owners of blocks.

Among the update-based systems, the LambdaNet multiprocessor exhibits the lowest average read latency, while the DMON-U system exhibits the highest. The average OPTNET read latency sits in between these two extremes. Discarding the Radix and Water results, the read latency in the LambdaNet system is only 7% shorter on average than in the OPTNET multiprocessor, while in the DMON-U multiprocessor reads are 20% more expensive on average than in the OPTNET system.

As seen in figure 7, the LambdaNet multiprocessor is usually more prone to contention effects than the OPTNET and DMON-U systems, due to two characteristics of the former system: a) its read and write transactions are not decoupled; and b) its absence of serialization points for updates from different nodes leads to an enormous update throughput. As a result of these characteristics, whenever an application involves an excessive amount of update traffic (Radix and Water being extreme cases), the read transactions are slowed down, as reads and writes compete for the same communication, cache, and main memory resources. Nevertheless, the performance degradation generated by contention is usually not enough to outweigh the very good base latencies in the LambdaNet system.

Contention affects the DMON-U and OPTNET systems in similar ways; both their contention-free and overall 2nd-level read miss latencies differ by 24% on average. Since contention-free 2nd-level read misses take longer to satisfy in the DMON-U system, this system exhibits worse read behavior than its OPTNET counterpart.

## 5.3 Performance of Writes

Having discussed the performance of read operations in each of the systems we study in the previous subsection, we move on to a study

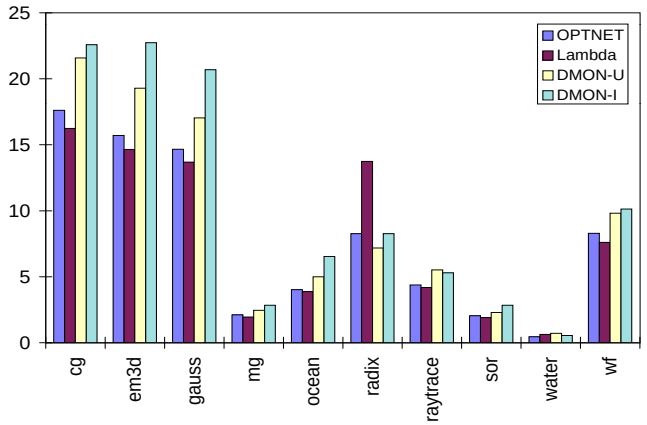



Figure 6: Average Read Latencies of (from left to right) OPTNET, LambdaNet, DMON-U, and DMON-I.

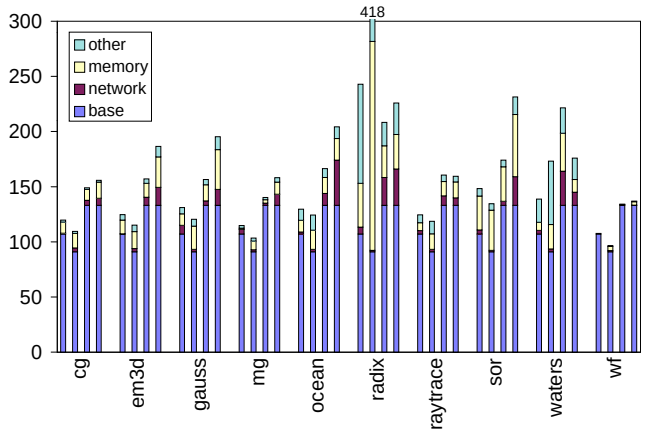



Figure 7: Average 2nd-Level Read Miss Latencies of (from left to right) OPTNET, LambdaNet, DMON-U, and DMON-I.

of the performance of write operations. Our results show that, except for Radix, the latency of write operations is negligible in all systems, showing that a 16-entry write buffer is usually enough to hide the overhead of coherence operations. In Radix writes are very frequent (roughly a rate of one write per 5 cycles) and cannot be coalesced in the write buffers, causing the buffers to stall the execution frequently. In addition, the write buffer flush overheads are negligible as a percentage of the overall execution time, even in the case of Radix. The only exception is Ocean running on DMON-I, where the write flush overhead represents 11.3% of the execution time.

These results suggest that the overhead of coherence operations is not a serious performance concern in most cases, even for the update-based systems which stress the communication system with a large number of updates. However, this is only the case because these systems include multiple coherence broadcast channels. Increasing the number of coherence channels has a significant impact on the medium access delay and on the amount of serialization imposed on coherence transactions by different nodes. As an example of this impact, consider a system with 16 nodes and a single TDMA coherence channel. In such a system, a node would be delayed an average 8 TDMA slots before getting access to the coherence channel. Furthermore, only one coherence transaction could be started during any slot. On the other hand, with two coherence channels, the same node would only be delayed an average 4 TDMA slots before starting its coherence transaction. Moreover, two coherence transactions could be started in parallel during any slot.

To quantify this effect in the case of the OPTNET system, consider figures 8 and 9. The figures show the running time of each of our applications on 16 and 32-node OPTNET systems, respectively, assuming 1, 2, and 4 coherence channels. The bars in the figure are broken down into busy time and read, write stall, and synchronization (including write buffer flush) overheads. All bars are normalized to the 1-channel results.

Two main observations can be made from these figures. The first is that performance can be significantly improved by using more than one coherence channel for several applications. Performance improvements come primarily from improvements in write performance, i.e. reduced write stall times and write buffer flush overheads. Note however that these improvements sometimes cause a significant increase in read latency, as in the cases of Radix and Water, as a result of increased contention.

The second observation is that two coherence channels are enough to get most of the benefit achievable by utilizing multiple channels, at least up to 32-node multiprocessors. Given that the gains achievable by increasing the number of update channels decrease exponentially, we believe that two coherence channels should deliver a better cost/performance ratio for machines with up to 64 or 128 nodes.

### 5.4 Impact of Architectural Parameters

We evaluated the impact of several of our simulation assumptions (the size of 2nd-level caches, the transmission rate, and the memory block read latency) in order to understand the behavior of the OPTNET architecture more fully. In summary, we find that these parameters do have a significant effect on performance. However, this effect is only quantitative, i.e. varying the parameters does not qualitatively change the trends observed and the outcome of the comparisons made in sections 5.1, 5.2, and 5.3. The complete study of the impact of our simulation assumptions is not presented here due to space limitations; more details can be found in [2].

## 6 Related Work

A common approach to using optical communication in computer networks is through WDM networks [3]. The use of this type of networks has become widespread as a result of recent advances in tunable transmitters and receivers and integrated optics technology. Optical networks with OTDM (Optical Time Division Multiplexing) have been proposed as an alternative to WDM networks, e.g. [8]. However, OTDM technology is not yet mature. Our work focuses on WDM technology due to its immediate availability, but nothing in OPTNET is strictly dependent on WDM.

Optical networks with WDM have been a part of other scalable parallel computer designs. Ghose *et al.* [5] proposed a WDM-based optical bus called Optimul to explore the benefits of the concurrent operation of multiple channels for both shared-memory and message-passing parallel computers. The architecture of Optimul is similar to that of the LambdaNet. Our comparison of OPTNET

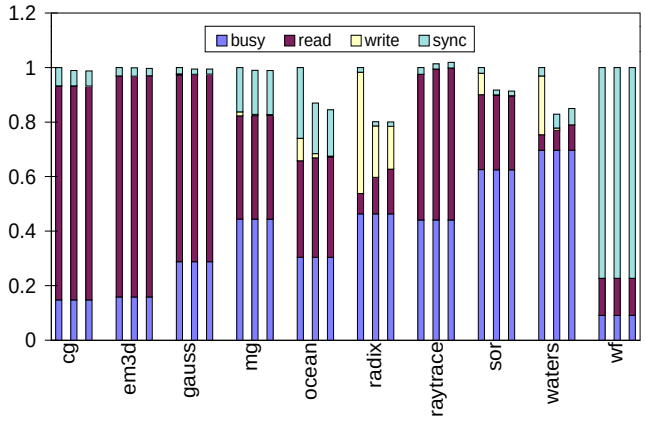



Figure 8: Run Times on 16-Node OPTNET System with (from left to right) 1, 2, and 4 Update Channels.

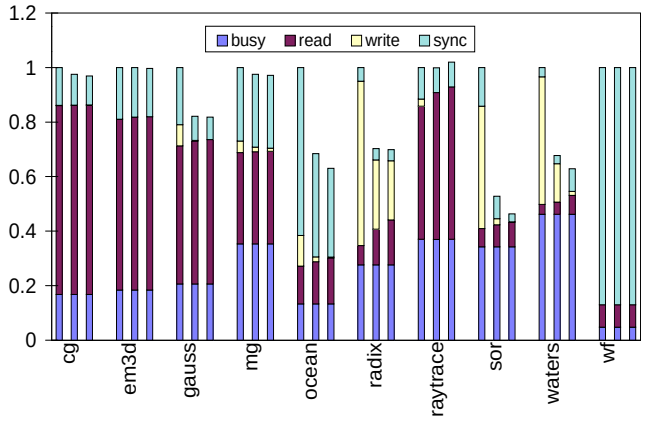



Figure 9: Run Times on 32-Node OPTNET System with (from left to right) 1, 2, and 4 Update Channels.

and LambdaNet-based systems indicates that our network should also have a better cost/performance ratio than Optimul, since: a) the performance differences between OPTNET and Optimul systems should be even smaller than between the OPTNET and LambdaNet systems; and b) the optical hardware cost of Optimul is only a constant factor better than that of the LambdaNet.

Ha and Pinkston [7] have proposed the DMON network and the DMON-I system studied in this paper. Our performance analysis has shown that the OPTNET multiprocessor outperforms the DMON-based systems in most cases. However, DMON-based systems have an advantage over OPTNET systems: they allow latency tolerance techniques based on multiple outstanding read requests; the OPTNET systems, as presented here, do not. This limitation results from the star coupler subnetwork having a single tunable receiver that must be tuned to a single home channel on a read access. Multiple outstanding read requests could be implemented on top of our network, if it were extended with a larger number of tunable receivers.

# 7 Conclusions

In this paper we proposed the OPTNET, a novel optical network and associated coherence protocol for scalable multiprocessors. Through a large set of detailed simulations, we showed that an OPTNET-based multiprocessor outperforms DMON-based systems consistently, even though the OPTNET requires no more hardware than DMON. In addition, a comparison between our system and a LambdaNet-based multiprocessor shows performance differences in the range of 0 to 12% in favor of the LambdaNet. We find this result to be extremely favorable to our system, given that the LambdaNet requires a factor of $p$ more hardware than the OPTNET, where $p$ is the number of nodes in the multiprocessor. Based on these results, we conclude that the combination of our network and coherence protocol strikes an excellent cost/performance ratio for multiprocessors.

## Acknowledgements

The authors would like to thank Timothy Pinkston and Joon-Ho Ha for their careful evaluation of our work and for discussions that helped improve this paper significantly.

# References

[1] D. Bailey *et al.* The NAS Parallel Benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, March 1994.

[2] E. V. Carrera and R. Bianchini. OPTNET: A Cost-Effective Optical Network for Multiprocessors. Technical Report Tech. Report ES-457/97, COPPE Systems Engineering, Federal University of Rio de Janeiro, December 1997.

[3] P. W. Dowd and J. Chu. Photonic Architectures for Distributed Shared Memory Multiprocessors. In *Proceedings of the 1st International Workshop on Massively Parallel Processing using Optical Interconnections*, pages 151–161, April 1994.

[4] K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. L. Hennessy. Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 15–26, May 1990.

[5] K. Ghose, R. K. Horsell, and N. Singhvi. Hybrid Multiprocessing in OPTIMUL: A Multiprocessor for Distributed and Shared Memory Multiprocessing with WDM Optical Fiber Interconnections. In *Proceedings of the 1994 International Conference on Parallel Processing*, August 1994.

[6] M. S. Goodman *et al.* The LAMBDANET Multiwavelength Network: Architecture, Applications, and Demonstrations. *IEEE Journal on Selected Areas in Communications*, 8(6):995–1004, August 1990.

[7] J.-H. Ha and T. M. Pinkston. SPEED DMON: Cache Coherence on an Optical Multichannel Interconnect Architecture. *Journal of Parallel and Distributed Computing*, 41(1):78–91, 1997.

[8] A. G. Nowatzyk and P. R. Prucnal. Are Crossbars Really Dead? The Case for Optical Multiprocessor Interconnect Systems. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 106–115, June 1995.

[9] J. E. Veenstra and R. J. Fowler. MINT: A Front End for Efficient Simulation of Shared-Memory Multiprocessors. In *Proceedings of the 2nd International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '94)*, 1994.

[10] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24–36, May 1995.