Speeding Up Multiprocessor Machines with Reconfigurable Optical Interconnects

W. Heirman[a], I. Artundo[b], L. Desmet[b], J. Dambre[a], C. Debaes[b], H. Thienpont[b], J. Van Campenhout[a]

[a]ELIS Department, Ghent University, Belgium
[b]TONA Department, Vrije Universiteit Brussel, Belgium

ABSTRACT

Electrical interconnection networks connecting the different processors and memory modules in modern large-scale multiprocessor machines are running into several physical limitations. In shared-memory machines, where the network is part of the memory hierarchy, high network latencies cause a significant performance bottleneck. Parallel optical interconnection technologies can alleviate this bottleneck by providing fast and high-bandwidth links. Moreover, new devices like tunable lasers and detectors or MEMS mirror arrays allow us to reconfigure the network at runtime in a data transparent way. This allows for extra connections between distant node pairs that communicate intensely, achieving a high virtual network connectivity by providing only a limited number of physical links at each moment in time. In this paper, we propose a reconfigurable network architecture that can be built using available low cost components and identify the limitations these components impose on network performance. We show, through detailed simulation of benchmark executions, that the proposed network can provide a significant speedup for shared-memory machines, even with the described limitations.

Keywords: Optical reconfiguration, Multiprocessor, Interconnection network, Distributed shared memory.

1. INTRODUCTION

In a modern large-scale multiprocessor machine, a high-speed electrical interconnection network connects different processors and memory modules over a distance of a few meters. At this combination of bandwidth and distance, several physical limitations become apparent.[1] In machines that provide a shared-memory paradigm, this network is part of the memory hierarchy.[2] Therefore, the ability to overlap memory access times with useful computation is severely limited by inter-instruction dependencies. Hence, high network latencies have a direct and often significant impact on performance.

It has been shown that optical interconnection technologies can alleviate this bottleneck.[3, 4] Mostly unhindered by crosstalk, attenuation and capacitive effects, these technologies will soon provide a faster, lower cost and more compact alternative to electrical interconnections, on distances from a few centimeters upward. Massively parallel inter-chip optical interconnects[5–7] are already making the transition from lab-settings to commercial products.

Optical signals may provide another advantage: the optical pathway can be modified by components like steerable mirrors, liquid crystals or diffractive elements. In combination with tunable lasers or photodetectors, these components will enable a runtime reconfigurable interconnection network[8, 9] that supports a much higher bandwidth than what is achievable through electrical reconfiguration technologies. From a viewpoint higher up in the system hierarchy, this would allow us to redistribute bandwidth or alter the network topology such that node-pairs that communicate intensely have a direct high-bandwidth, low-latency connection.

However, the switching time for most of these components is such that reconfiguration will necessarily take place on a time scale that is significantly above that of the individual memory accesses. The efficiency with which such networks can be deployed will therefore strongly depend on the temporal behavior of the data

Further author information:
Wim Heirman: E-mail: wim.heirman@elis.ugent.be, Telephone: +32 (0)9 264 95 27
Iñigo Artundo: E-mail: iartundo@tona.vub.ac.be, Telephone: +32 (0)2 629 18 14

transfer patterns. We have characterized the locality in both time and space of the traffic flowing over the network in previous studies,[10] using large-scale simulations of the execution of real benchmark programs with a simulation platform based on the Simics multiprocessor simulator.[11] We have found that long periods of intense communication occur between node pairs suggesting that slowly reconfiguring networks can result in a significant application speedup. Subsequently we have included the model of a specific reconfigurable network in our simulator, enabling us to measure its effect on the performance of the machine.

In this paper, we will propose an implementation of this reconfigurable network. First we provide an overview of both the architecture of the distributed shared-memory machine that is used throughout the study, and of the abstract model of the reconfigurable network on which our simulations are based. Next we give an overview of the optical components that can be used to implement this network model. These include tunable VCSELs, a Selective Optical Broadcast (SOB) device and wavelength specific photodetectors.

We will identify how some of the properties of these components affect high-level network parameters, and translate these into additional limitations that are imposed on our network model. Three limitations are identified. Component count and cost limit the fan-out of our network. The tuning range of the transmitters, combined with the need for efficient use of transmitted power, forces us to partition the network using the SOB device, this restricts the topology of the reconfigurable part of the network. Finally the tuning speed of the VCSELs affects the reconfiguration speed of the network. For each of these limitations a number of simulations were done with different parameters. This way, we can determine the effect on the performance of the complete machine for each of the limitations separately, and determine which of them should be the most likely candidate for future improvement.

This paper is structured as follows. In section 2, we describe the architecture of the shared-memory machine and the topology of the reconfigurable network as they are used in our simulations. Section 3 gives an overview of the optical components that can be used to provide the reconfigurability, and relate properties of the components to limitations on our network model. Next we report on our experiments, with section 4 detailing the multiprocessor simulation platform we used, and section 5 providing measurements of the application speedup, taking each of the limitations into account. Finally, we propose some future work in section 6 and summarize our conclusions in section 7.

2. SYSTEM ARCHITECTURE

2.1. Multiprocessor architecture

Multiprocessor machines come in two basic flavors: those that have a tight coupling between the different processors and those with a more loose coupling. Both types can conceptually be described as consisting of a number of nodes, each containing a processor, some memory and a network interface, with a network connecting the different nodes (figure 1). In the extreme end of the loosely coupled family we find examples such as the *Beowulf cluster*,[12] in which the network consists of a commodity technology such as Ethernet. This simplistic interconnection network, connected to the processors through several layers of I/O-busses, results in relatively low throughput (1 Gbps per processor) and high latency (several hundred microseconds). These machines are necessarily programmed using the message passing paradigm, and place a high burden on the programmer to efficiently schedule computation and communication.

On the other hand, tightly coupled machines usually have proprietary interconnection technologies that are situated at a much lower architectural level (much closer to the processors), resulting in higher throughput (tens of Gbps per processor) and very low latency (down to a few hundred nanoseconds). This makes them suitable for solving problems that can only be parallelized into tightly coupled subproblems (i.e., that communicate often). It also allows them to implement a hardware-based shared-memory model, in which communication is initiated when a processor tries to access a word in memory that is not on the local node, *without programmer's intervention.* This makes shared-memory based machines relatively easy to program. Since the network is now part of the memory hierarchy, it also makes such machines much more vulnerable to increased network latencies.

Modern examples of this last class of machines range from small, 2- or 4-way SMP server machines, over mainframes with tens of processors (Sun Fire, IBM iSeries), up to supercomputers with hundreds of processors (SGI Altix, Cray X1). The larger types of these machines are already interconnect limited, and since the

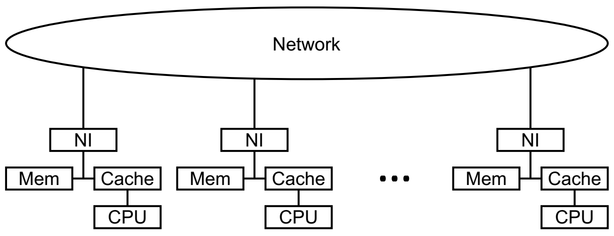

Figure 1. Schematic overview of a multiprocessor machine. For message-passing machines, the network traffic is under control of the application. In shared-memory machines, network traffic is generated by the network interfaces (NI) in response to non-local memory accesses by a processor.

capabilities of electrical networks are evolving much more slowly than processor frequencies, they make very likely candidates for the application of reconfigurable optical interconnection networks.

For this study we consider a machine in which coherency is maintained through a directory based coherency protocol. This protocol was pioneered in the Stanford DASH multiprocessor,[2] and is, in one of its variants, used in all modern large shared-memory machines. In this computing model, every processor can address all memory in the system. Accesses to words that are allocated on the same node as the processor go directly to local memory; accesses to other words are handled by the network interface. This interface will generate the necessary network packets requesting the corresponding word from its home node. Since processors are allowed to keep a copy of remote words in their own caches, a cache coherency protocol has to be implemented that keeps all copies synchronized. The network interfaces keep a directory of which processor has which word in its cache, and make sure that, before a processor is allowed to write to a word, all copies of the same word in the caches of other processors are invalidated. Network traffic thus consists of both control information (read/write/invalidate requests and acknowledgements) and data traffic (words that are sent to the caches or written back to main memory). In this setup, one memory access can take the time of several network round trips (hundreds of nanoseconds). This is much more than the time out-of-order processors can occupy with other, non-dependent instructions, but not enough for the operating system to schedule another thread. This makes it very difficult to effectively hide the communication latency, and makes the system performance very much dependent on network latency.

2.2. A reconfigurable network architecture

Previous studies concerning reconfigurable networks have mainly dealt with fixed topologies (usually a mesh or a hypercube) that allowed swapping of node pairs, incrementally evolving the network to a state in which processors that often communicate are in neighboring positions.[13, 14] However, algorithms to determine the placement of processors turned out to converge slowly, or not at all when the characteristics of the network traffic change rapidly – this is the case with most real-world applications.

Therefore, we assume a different network architecture in this study. We start from a base network with a fixed, regular topology. In addition, we provide a second network that can realize a limited number of connections between arbitrary node pairs – these will be referred to as *extra links* or *elinks*. A schematic overview is given in figure 2. An advantage of this setup, compared to other topologies that allow for more general reconfiguration, is that the base network is always available. This is most important during periods where the extra network is undergoing reconfiguration and can not be used. Routing and reconfiguration decisions are also simplified because it is not possible to completely disconnect a node from the others – all nodes are at all times connected through the base network.

The extra links are positioned such that node pairs that exchange the most data are 'close together': the distance, defined as the number of hops a packet sent between the node pair must traverse in the resulting network topology, is minimized. This way, a large percentage of the traffic has a short path which avoids arbitration and possibly buffering at intermediate stages. Also congestion is lowered because heavy traffic is no longer spread

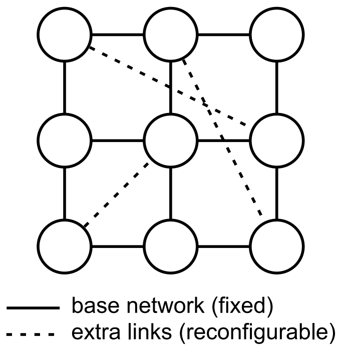

Figure 2. Reconfigurable network topology. The network consists of a static base network with regular topology, augmented with a limited number of direct, reconfigurable 'extra' links.

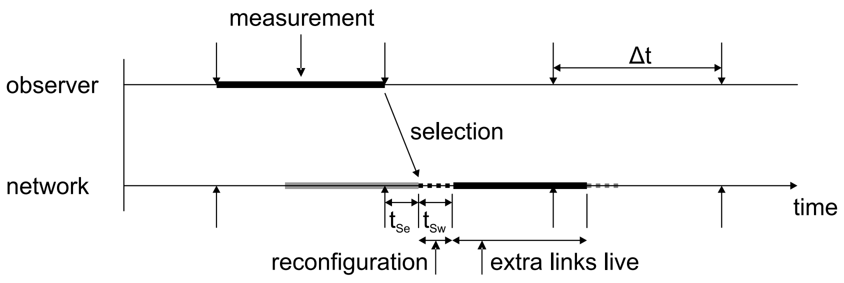

Figure 3. The observer measures network traffic, and after each interval of length Δt makes a decision where to place the extra links. This calculation takes an amount of time called the *selection time* (t_{Se}). Reconfiguration will then take place, during which time the extra links are unusable (*switching time*, t_{Sw}).

out over a large number of links. Traffic remaining on the base network therefore experiences less contention and is also accelerated.

Since the network traffic changes over time, we need to revise the position of the extra links periodically. Therefore, we reconfigure the network at specific intervals, the length of each interval being a (fixed) parameter of the network architecture (the 'reconfiguration interval', denoted by Δt, see figure 3). Traffic is observed by a reconfiguration entity during the course of an interval, and total traffic between each node pair is computed. At the end of the interval the new positions of the extra links are determined (based on the traffic matrix measured during the past interval, using the algorithm described in section 4.2) and the network configuration is updated accordingly. Computing the new configuration takes time: the *selection time* (t_{Se}). Its duration depends on the complexity of the selection algorithm and the method of implementation (in hardware or software). In addition, the reconfiguration itself is not immediate: depending on the technology used, reconfiguration can take from 10 μs up to several ms, this is the *switching time* (t_{Sw}). For our simulations, we assumed that the computation to select the new elinks and the physical reconfiguration (the tuning of the lasers) each take up 10% of the reconfiguration interval.

To amortize on the cost of reconfiguration, during which time the extra links are inactive, the reconfiguration interval should be chosen significantly longer than the switching time. On the other hand, if the reconfiguration interval is too long, the elink placement for one interval, based on traffic measurements from the *previous* interval, will not be a good match for traffic in the current interval, degrading the performance improvement obtained.

This simple model provides us with a unified, parameterized architecture that allows us to analyze and simulate several types of network implementations. A real network will of course impose some physical limitations, these are introduced in section 3.

3. OPTICAL COMPONENTS

3.1. Overview

Three major approaches for optical reconfiguration are the use of active tunable opto-electronics, functionally tunable optical components, or beam steering micro-optical electro-mechanical systems (MOEMS).

In tunable opto-electronics the wavelength or polarization state of the emitted light can be changed by controlling the temperature, the current injection or by changing the micro-cavity geometry with micro-electro-mechanical (MEMS) membranes. Some of the more remarkable achievements in this field are the advent of MEMS based 2D tunable Vertical-Cavity Surface-Emitting Laser (VCSEL) arrays and Arrayed Waveguide Grating (AWG) based tunable fiber lasers.[15] Tunable opto-electronic components can be used in a passive optical broadcast-and-select scheme where the wavelength is selecting the destination. The broadcast can be achieved in a guided wave approach through passive star couplers or by beam splitting diffractive optical elements (DOEs) in the free-space case. The select mechanism can be through the use of resonant cavity photodetectors (RCPDs), tunable optical filters, AWGs, passive polarization sensitive DOEs[16] or passive wavelength sensitive DOEs. In general, these special DOEs generate different fan-out patterns for changed wavelength or polarization conditions of the incident light. The most recent examples of wavelength selective devices are microring waveguide resonators.[17] They are very good candidates for photonic integration since high compactness and low-cost can be achieved in the silicon on insulator (SOI) technology.

Another method for optical switching is to tune the functionality of the optical components themselves instead of tuning the characteristics of the light sources in the network. In the past, tuning has been achieved by acousto-optic Bragg cells,[18] with photorefractive crystals,[19] by heating liquids in thermal bubble switches[20] or by using micro-fluids as the tuning element in waveguides.[21] Recently, Liquid Crystal (LC) based optical components also receive a lot of attention. LCs can be used to make adaptive computer generated holograms (CGHs) and switchable gratings.[22]

A third approach for optical switching is the use of active free-space laser beam steering. Here, a MEMS micromirrors based chip device images a 2D fiber array onto a second one.[23] These devices are compact, consume little power and can be batch fabricated resulting in low cost. However, MEMS optical crossconnects have yet to overcome design challenges in reliable opto-mechanical packaging, mirror fabrication and complicated electronics and control algorithms.

3.2. Reconfigurable network implementation

Contrary to long-range communications networks like metropolitan- and wide-area networks, short-range (no more than a few meters) interconnection networks try to avoid complicated switching devices or costly opto-electronics. This rules out some of the components described above, for instance the AWG lasers and most other lasers with sub-microsecond tuning speed. If possible, we would even like to have links that can transmit a single packet over multiple fibers in parallel, since this greatly reduces packet latency (note that latency is the most important parameter for networks in this setting, bandwidth is only important to avoid congestion since congestion introduces additional latency). Therefore, we believe that systems which rely on coarse wavelength tunability are a viable approach for introducing reconfiguration at reasonable costs. Driven by the progress in metro-access networks, low-cost tunable devices such as MEMS-based tunable VCSELs[24] are becoming readily available.

The reconfigurable part of the inter-processor communication network could consist of a single tunable optical transmitter per processor node[*]. This way, each node can transmit data on a fixed number of wavelengths. This signal is then guided to a broadcasting element which divides the data-carrying signal to all (or a selection) of the receiving nodes. Each processor node also incorporates an optical receiver which is sensible to one wavelength only. Hence, by tuning the wavelength of each transmitter, a receiving node can be chosen, altering the effective topology of the network.

VCSELs are ideal candidates for optical interconnects,[25] because they can be mass produced on wafer-scale, they exhibit low power consumption, and their rotationally symmetrical laser beam can easily be coupled into

[*]Or an array of them, if parallel transmission is desired. Note that in this case the complete array is part of *one* link, so all should be tuned to the same wavelength.

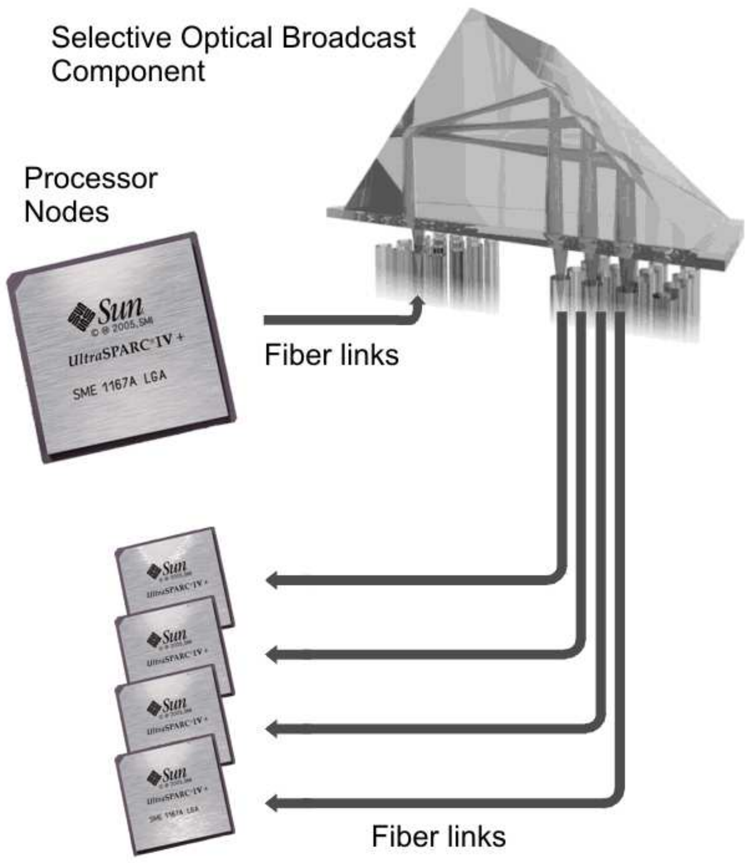

Figure 4. Schematic representation of the broadcast from one processor node in the proposed reconfigurable network architecture (relative component sizes are not to scale). The top processor node transmits data on one of four wavelengths $\lambda_1 \ldots \lambda_4$. The optical broadcast element distributes the signal to four processor nodes. Since every receiving node is sensitive to one wavelength only, the target node is selected by emitting at the appropriate wavelength.

optical fiber while offering a small form factor for easy onboard integration. They can also easily be fabricated in 1- or 2-D arrays to provide parallel links. At the receiving part of the processor node, a resonant cavity photodetector (RCPD) can be used. This type of photodetectors is only sensitive to a very narrow band of optical frequencies. Their wavelength selectivity is enhanced by a Fabry-Perot cavity.

A broadcast-and-select scheme in which all processing nodes (say over 64 nodes) are connected together via a single star-coupling element is not a scalable design. The channel count of the VCSELs is limited, also broadcasting to a large number of nodes of which only one is the actual receiver is an inefficient use of transmitted power. We therefore propose using a selective optical broadcasting (SOB) component[26] which broadcasts each incoming channel to a limited number of outputs. Figure 4 shows the concept of the proposed reconfigurable network containing such a passive optical broadcast element. The fibers coming from the tunable sources in each processing node are bundled into a fiber array at the ingress of the SOB device. This free-space optical component will fan-out the optical signal to a 3x3 matrix of spots at the output side via a diffraction grating on its sides. This way, each node is capable of addressing 9 different receiver nodes. A scalable reconfigurable interconnection scheme is thus possible using transmitters with low channel counts.

It is important to note that the mapping of the source node connections to the receiving nodes on the broadcasting component is now critical, because it directly determines the possible addressable nodes for every transmitting node. We have determined an optimal placement, based on the following considerations. The latency of a packet across the network is a function of the hop distance, i.e. the minimal number of intermediate nodes (hops) the packet must traverse before reaching its destination. On average, it is most beneficial for packet latency to create extra optical links between node pairs that are far apart in the base network topology. An optimal topology is hence one that has minimal hop distance among all node pairs. With the proposed selective broadcasting scheme, 9 additional nodes can be potentially reached in a single hop for every extra link to the SOB device. At runtime however, only one of these 9 nodes can actually be addressed since the transmitter on this node is tuned to the frequency of only one of them. This means that for reconfigurable networks the hop distance will vary over time. Therefore we have calculated the 'potential hop distance' (the minimal hop distance over all reconfiguration possibilities) for every node pair as a metric for evaluating the relative performance of different placements.

We have implemented a simulated annealing algorithm to determine the best possible placement matrix in terms of total potential hop distance (summed over all node pairs). For the 16-node torus network and the configuration with broadcast towards 9 neighboring nodes, our best placement results in a potential hop distance improvement of 40.8% compared to the average of 1000 random placements. This improvement increases as the number of nodes in the network increases. Performing the same optimization for larger torus networks resulted in more pronounced potential hop distance reductions, with an asymptotic value of 60% for very large networks.

3.3. Network limitations

In 2.2, we introduced our abstract network model. This model assumed that the reconfigurable part of the network could establish a fixed number (denoted by n) of extra links between arbitrary node pairs (with N the number of nodes). Yet the physical implementation in 3.2 does not support such a general topology. However, by imposing some constraints on the placement of the extra links we can express these physical implementation within our abstract model.

First, there is only one reconfigurable transmitter and one wavelength-selective receiver per node. This means that only one extra link can terminate at each node. We incorporate this into the model by imposing a *fan-out* of $f = 1$. This also fixes the number of extra links that are active at the same time to equal the number of nodes: $n = N$.

Second, when an extra link has a given node at one endpoint, the choice for the second endpoint is limited. Indeed, only 9 different nodes can be reached through the SOB device. Therefore, the selection of active extra links should be made out of only $9 \cdot N$ different node pairs instead of $N \cdot (N - 1)$ in the general case.

Finally, the switching time (t_{Sw}) of the VCSELs provides a lower bound on the reconfiguration interval. During switching the extra link cannot be used, so the reconfiguration interval must be significantly higher than the switching time: $\Delta t \gg t_{Sw}$.

In the next sections, we will report on our simulations, which try to answer whether a reconfigurable network adhering to these limitations can still provide a significant benefit. We will also see which of these limitations causes the greatest degradation in performance, and how this could be avoided.

4. SIMULATION SETUP

4.1. Simulation platform

We have based our simulation platform on the commercially available Simics simulator.[11] It was configured to simulate a multiprocessor machine resembling the Sun Fire 6800 server, with 16 UltraSPARC III processors clocked at 1 GHz and running the Solaris 9 operating system. Stall times for caches and main memory are set to realistic values (2 cycles access time for L1 caches, 19 cycles for L2 and 100 cycles for SDRAM). The directory-based coherency controllers and the interconnection network are custom extensions to Simics, and model a full bit vector directory-based MSI-protocol and a packet-switched 4x4 torus network with contention and cut-through routing. For our simulations, a number of extra point-to-point links can be added to the torus topology at any point in the simulation.

The network links in the base network provide a bandwidth of 1.6 Gbps and a per-hop latency of 10 ns. In the reported experiments, the characteristics of an extra link were assumed to be equal to those in the base network, yielding the same latency for an extra link as for a single base network link. Both coherency traffic (read requests, invalidation messages etc.) and data (the actual cache lines) are sent over the network. The resulting remote memory access times are representative for a Sun Fire server (around 1 μs on average).

To avoid deadlocks, dimension routing is used on the base network. Each packet can go through one extra link on its path, after that it switches to another virtual channel (VC)[†] to avoid deadlocks of packets across extra links. For routing packets through the network, each node has a static routing table that provides the next step in the shortest path to each destination node. For each destination, this table tells the node to route packets either through an elink starting at that node, to the start of an elink on another node, or straight to its destination, the latter two using the base network with normal dimension routing. When reconfiguring the network, all routing tables are updated.

Since the simulated caches are not infinitely large, the network traffic will be the result of both coherency misses and cold/capacity/conflict misses. To make sure that private data transfer does not become excessive, a first-touch memory allocation was used that places data pages of 8 KiB on the node of the processor that first references them. Also each thread is pinned down to one processor (using the Solaris `processor_bind()` system call), so the thread stays on the same node as its private data for the duration of the program.

The SPLASH-2 benchmark suite[28] was chosen as the workload. It consists of a number of scientific and technical applications and is a good representation of the real-world workload of large shared-memory machines. Because the default benchmark sizes are too big to simulate their execution in a reasonable time, smaller problem sizes were used. Since this influences the working set, and thus the cache hit rate, the level 2 cache was resized from the 8 MiB on a real UltraSPARC III to 512 KiB, resulting in an 80% L2 hit rate.

4.2. Network architecture

The interconnection network in our simulator implements the model described in section 2.2, with the number of extra links fixed at $n = 16$. We gradually added the additional constraints from section 3.3: a limited fan-out, the restriction on the node pairs as prescribed by the placement matrix of the SOB, and finally we sweep over a range of reconfiguration intervals. Selection time t_{Se} and switching time t_{Sw}, both denoted on figure 3, are both set to 10% of the reconfiguration interval Δt.

When determining extra link placements, we want to minimize the number of hops (this is the number of intermediate nodes a packet should pass from source to destination) for most of the traffic. Formally, our cost function can be expressed as

$$C = \sum_{i,j} d(i,j) \cdot T(i,j)$$

with $d(i,j)$ the hop distance between nodes i and j, which is a function of the elinks that are selected to be active, and $T(i,j)$ the traffic (in number of bytes) exchanged between the node pair in the time interval of interest. Since the reconfiguration interval will typically be in the order of milliseconds, and the extra link selection should be done in only a fraction of this time, we need a fast heuristic that can quickly find a set of active elinks that satisfies the constraints imposed by the architecture and has an associated cost close to the global optimum.

We have developed a greedy algorithm that traverses a list of node pairs, sorted by $d(i,j) \cdot T(i,j)$ in descending order. In each step, we will activate at most one extra link. First we look at which elinks are still available, this means that they can be activate without violating the limitation imposed on the fan-out f. Out of these available elinks, the *most interesting* one is activate. This is the elink that provides the greatest reduction in distance, relative to the distance over the base network *and* already active extra links. If none of the available links provides a reduction in distance, no new link is added for this node pair. This process is continued for the next node pair on the list until there are no more available elinks. For performance reasons, the hop distance $d(i,j)$ used to sort the node pairs uses only base network links and is not recomputed every time a new elink is selected.

Note that this algorithm by no means provides the global optimum for the extra link selection. The greedy approach was chosen since it is very fast – unless very long reconfiguration intervals are used, we only have

[†]Actually another *set* of VCs is used since we already employ separate request and reply VCs to avoid *fetch deadlock*[27] at the protocol level.

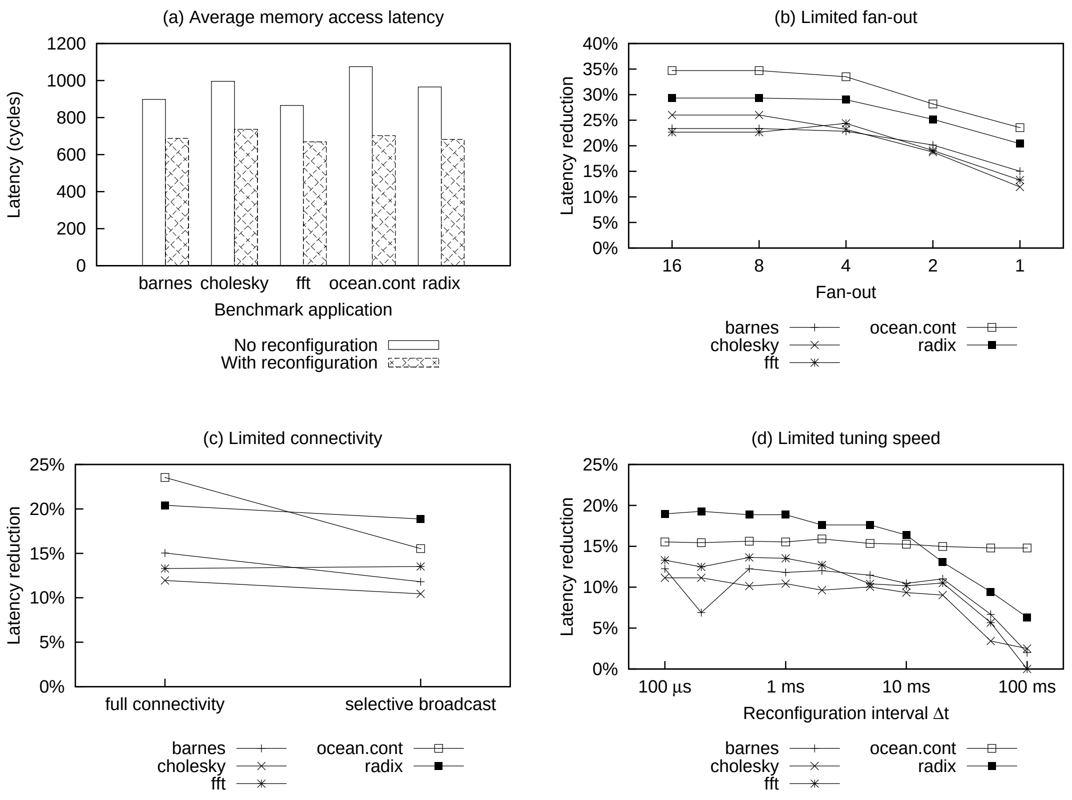

Figure 5. Results of our simulations. (a) shows the average remote memory access latency, with and without an (idealized) reconfigurable network model with 16 extra links and a reconfiguration interval of 1 ms; (b) shows the improvement of the access latency after adding reconfiguration while reducing the maximum number of extra links terminating at any one node; in (c) we constrained the placement of the extra links to those possible using the SOB device; and (d) shows the effect of different reconfiguration intervals on a selective broadcast network.

less than a millisecond to do this selection. We also computed the global optimum for a large number of traffic patterns, the cost of a solution provided by the greedy algorithm was always under 10% more than this global optimum.

5. RESULTS

Figure 5 summarizes all our simulation results. In figure 5(a), we show the average remote memory access latency for a selection of the SPLASH-2 benchmarks, first using only the 4x4 torus (base) network, next when a reconfigurable network is added. This is done using the idealized network model as described in section 2.2, with 16 extra links, a reconfiguration interval of 1 ms and no constraints on which nodes can be interconnected or how many links may terminate at any one node. The access latency is significantly reduced, from 960 clock cycles to only 696 cycles (averaged over the 5 benchmarks), this is a reduction of 27%. The next graphs show what remains of this improvement, once we transform our idealized network model into something that can be physically implemented by introducing the constraints derived in section 3.3.

First, we account for the fact that only one reconfigurable transmitter and one wavelength-selective receiver are available per node. This means that only one extra link can terminate at each node. Figure 5(b) shows the reduction of memory access latency over the baseline (using only the base network), when we gradually reduce the maximum fan-out from $f = 16$ (there are only 16 extra links, so this effectively means no fan-out limitation) to just one extra link per node. The reconfiguration interval is fixed at 1 ms. Usually, network traffic is 'localized', which means most nodes only talk to a limited set of other nodes. This is visible in the graph: we can reduce the fan-out to 4 almost without sacrificing performance. After that, the network is unable to provide speedup for all of the memory accesses, and latency increases again (latency reduction goes down). Still, half to two-thirds of the reduction is maintained after limiting fan-out to one. By placing two or more tunable lasers and receivers per node, one could implement a network with higher performance. This would off course drive up the cost. According to figure 5(b), using more than four would not provide additional gain.

Second, we introduce the SOB device into our simulations. This limits the choice of possible node pairs connected with an extra link from $N \cdot (N-1)$ to just $9 \cdot N$. If two nodes communicating heavily are not in the list of $9 \cdot N$ possible node pairs, there cannot be a direct link between them so we expect performance would suffer from this restricted connectivity. As figure 5(c) shows, this is only the case to a limited extent. Indeed, even though a node pair cannot have a direct connection, it may still be possible to offer a shorter path using one extra link and one base network link, compared to 3 or 4 base network links when no extra links are available. Also, connecting 2 nodes with slightly less traffic between them with an elink can free up part of the base network for use by other heavy traffic streams. This makes that a reconfigurable network implementation using the SOB device can still provide a significant performance improvement, while being scalable to a large number of nodes.

Finally, we look at how the reconfiguration interval affects performance. Since traffic is expected to change over time, we would like to reconfigure the network as often as possible. Remember however that the reconfiguration interval is set to ten times the switching time of the VCSELs, a shorter interval would mean that the extra links are unavailable for more than 10% of the time, limiting their usefulness. Figure 5(d) shows the latency reduction for a number of reconfiguration intervals, using the reconfigurable network implementation with the SOB device. If VCSELs with a tuning speed of 1 ms can be used, the reconfiguration interval would be 10 ms which seems to be the turning point for most of the benchmark applications. Note that the applications differ most in how fast their traffic patterns change, so here the results are more differentiated between the different benchmarks compared to the previous graphs.

Overall, we can conclude that limiting fan-out to one and using selective instead of full broadcast are necessary to allow the network to be implemented at a reasonable cost, but they reduce the access latency improvement to about half of what we could achieve if these limitations were not present. With a reconfiguration interval of at most 10 ms, requiring a VCSEL tuning time of about 1 ms, no additional performance is sacrificed.

6. FUTURE WORK

The research on optical components is moving ahead. Other implementations of our reconfigurable architecture are possible than the one described here, several more will be possible in the near future. We will follow these developments and update our simulation models accordingly, and report on their performance. Furthermore we will take our simulations beyond what is possible today and explore how future implementations could provide even better results, translating this into recommendations on how new components could be developed that are a good match to our application domain.

7. CONCLUSIONS

We have introduced a reconfigurable network architecture that can be used to speed up remote memory accesses in a large shared-memory multiprocessor machine. We explored how this network can be implemented using reconfigurable optical components that are available in our labs or on the market, and how the properties of these components relate to constraints on the high-level network model. Finally, we provided simulation results that characterize the performance of our reconfigurable network over a range of network parameters. These results show that a significant reduction in memory access latency can be achieved even with the limitations the components impose on our network model. The latency reduction that can be expected for the specific shared-memory machine that was modeled has been shown to be between 10 and 20%, depending on the benchmark, when VCSELs with a tuning time of 1 ms or less are available.

ACKNOWLEDGMENTS

This paper presents research results of the PHOTON Inter-university Attraction Poles Program (IAP-Phase V), initiated by the Belgian State, Prime Minister's Service, Science Policy Office. C. Debaes is indebted to the FWO for his post-doctoral fellowship.

REFERENCES

1. D. A. B. Miller and H. M. Ozaktas, "Limit to the bit-rate capacity of electrical interconnects from the aspect ratio of the system architecture," *Journal of Parallel and Distributed Computing* **41**(1), pp. 42–52, 1997.
2. D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. L. Hennessy, M. Horowitz, and M. S. Lam, "The Stanford DASH multiprocessor," *IEEE Computer* **25**, pp. 63–79, March 1992.
3. J. Collet, D. Litaize, J. V. Campenhout, M. Desmulliez, C. Jesshope, H. Thienpont, J. Goodman, and A. Louri, "Architectural approach to the role of optics in monoprocessor and multiprocessor machines," *Applied Optics* **39**, pp. 671–682, 2000.
4. A. F. Benner, M. Ignatowski, J. A. Kash, D. M. Kuchta, and M. B. Ritter, "Exploitation of optical interconnects in future server architectures," *IBM Journal of Research and Development* **49**(4/5), pp. 755–776, 2005.
5. M. Brunfaut *et al.*, "Demonstrating optoelectronic interconnect in a FPGA based prototype system using flip chip mounted 2D arrays of optical components and 2D POF-ribbon arrays as optical pathways," in *Proceedings of SPIE*, **4455**, pp. 160–171, (Bellingham), July 2001.
6. L. Chao, "Optical technologies and applications," *Intel Technology Journal* **8**, May 2004.
7. L. Schares *et al.*, "Terabus – a waveguide-based parallel optical interconnect for Tb/s-class on-board data transfers in computer systems," in *Proceedings of the 31st European Conference on Optical Communication (ECOC 2005)*, **3**, pp. 369–372, The Institution of Electrical Engineers, (Glasgow, Scotland), September 2005.
8. W. Heirman, I. Artundo, D. Carvajal, L. Desmet, J. Dambre, C. Debaes, H. Thienpont, and J. Van Campenhout, "Wavelength tuneable reconfigurable optical interconnection network for shared-memory machines," in *Proceedings of the 31st European Conference on Optical Communication (ECOC 2005)*, **3**, pp. 527–528, The Institution of Electrical Engineers, (Glasgow, Scotland), September 2005.
9. C. Katsinis, "Performance analysis of the simultaneous optical multi-processor exchange bus," *Parallel Computing* **27**(8), pp. 1079–1115, 2001.
10. W. Heirman, J. Dambre, J. Van Campenhout, C. Debaes, and H. Thienpont, "Traffic temporal analysis for reconfigurable interconnects in shared-memory systems," in *Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium*, p. 150, IEEE Computer Society, (Denver, Colorado), April 2005.
11. P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *IEEE Computer* **35**, pp. 50–58, February 2002.
12. T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer, "Beowulf : A parallel workstation for scientic computation," in *Proceedings of the International Conference on Parallel Processing, Boca Raton, USA*, pp. 11–14, CRC Press, August 1995.
13. T. M. Pinkston and J. W. Goodman, "Design of an optical reconfigurable shared-bus-hypercube interconnect," *Applied Optics* **33**(8), pp. 1434–1443, 1994.
14. J. L. Sánchez, J. Duato, and J. M. García, "Using channel pipelining in reconfigurable interconnection networks," in *6th Euromicro Workshop on Parallel and Distributed Processing*, January 1998.
15. S. Liaw *et al.*, "Wavelength tuning and multiple-wavelength array-waveguide-grating lasers," *Optical Engineering* **12**, pp. 2178–2179, August 2003.
16. W. Peiffer, H. Thienpont, M. Pelt, and I. Veretennicoff, "Polarisation-controlled module for reconfigurable nearest neighbour optical interconnects and image shifting," *International Journal of Optoelectronics* **9**(3), p. 273, 1994.
17. B. E. Little, S. T. Chu, W. Pan, and Y. Kokubun, "Microring resonator arrays for VLSI photonics," *IEEE Photonics Technology Letters* **12**, pp. 323–325, 2000.
18. E. Tervonen, A. T. Friberg, J. Westerholm, J. Turunen, and M. R. Taghizadeh, "Programmable optical interconnections by multilevel synthetic acousto-optic holograms," *Optics Letters* **16**(16), pp. 1274–1276, 1991.
19. A. Chiou and P. Yeh, "2 x 8 photorefractive reconfigurable interconnect with laser diodes," *Applied Optics* **31**(26), pp. 5536–5541, 1992.

20. J. Yang, Q. Zhou, and R. T. Chen, “Polyimide-waveguide-based thermal optical switch using total-internal-reflection effect,” *Applied Physics Letters* **81**(16), pp. 2947–2949, 2002.
21. V. Lien, Y. Berdichevsky, and Y.-H. Lo, “A prealigned process of integrating optical waveguides with microfluidic devices,” *IEEE Photonics Technology Letters* **16**, pp. 1525–1527, 2004.
22. Y. Boiko, J. Eakin, J. Vedrine, and G. P. Crawford, “Polarization-selective switching in holographically formed polymer dispersed liquid crystals,” *Optics Letters* **27**, p. 1717, 2002.
23. A. Neukermans and R. Ramaswami, “MEMS technology for optical networking applications,” *IEEE Communications Magazine* **39**, pp. 62–69, January 2001.
24. A. Filios *et al.*, “Transmission performance of a 1.5 μm 2.5 Gb/s directly modulated tunable VCSEL,” *IEEE Photonics Technology Letters* **15**, 2003.
25. R. Michalzik, J. Ostermann, M. Riedl, F. Rinaldi, H. Roscher, and M. Stach, “Novel VCSEL designs for optical interconnect applications,” in *The 10th OptoElectronics and Communications Conference (OECC 2005)*, (Seoul, Korea), 2005.
26. C. Debaes *et al.*, “Low-cost micro-optical modules for MCM level optical interconnections,” *IEEE Journal of Selected Topics in Quantum Electronics, Special issue on Optical Interconnects* **9**, pp. 518–530, January 2003.
27. C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong-Chan, S.-W. Yang, and R. Zak, “The network architecture of the Connection Machine CM-5,” *Journal of Parallel and Distributed Computing* **33**(2), pp. 145–158, 1996.
28. 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 22th International Symposium on Computer Architecture*, pp. 24–36, (Santa Margherita Ligure, Italy), 1995.