Designing Communication Strategies for Heterogeneous Parallel Systems *

Ravi Prakash **Dhabaleswar K. Panda**
Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210
Tel:(614)-292-5199, Fax: (614)-292-2911
E-mail: {prakash, panda}@cis.ohio-state.edu

Contact Author: Dr. Dhabaleswar K. Panda

Abstract

This paper investigates communication strategies for interconnecting heterogeneous parallel systems. As the speed of processors and parallel systems keep on increasing over the years, electronic interconnections like HIPPI and FDDI are reaching their limit to provide heterogeneous parallel computing environment. This paper explores the suitability of the emerging passive star-coupled optical interconnection using wavelength division multiplexing as the system interconnect to provide high bandwidth (Gbits/sec) communication demanded by heterogeneous systems. Several different communication strategies (combinations of communication topologies and protocols) over *Wavelength Division Multiplexed* (WDM) communication media like optic fiber are investigated under a representative master-slave computational model. The interplay between system speed, network speed, task granularity, and degree of parallelism is studied using both analytical modeling and simulations. It is shown that a hierarchical ALOHA-based communication strategy between the master and the slaves, implemented on top of the passive star-coupled network, leads to a considerable reduction in channel contention and provides $50\% - 80\%$ reduction in task completion time for applications with medium to high degrees of coarse grain parallelism. Comparable reduction in channel contention is also shown to be achieved by using tunable acoustooptic filters at master nodes.

Keywords: Parallel computer architecture, heterogeneous parallel systems, hierarchical interconnection, optic fiber networks, and meta-parallelism.

* A preliminary version of this paper [26] has been presented and received an *Outstanding Paper Award*, at the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks, Kanazawa, Japan, December 14–16, 1994. This research is supported in part by the National Science Foundation Grant # MIP-9309627.

Contents

1 Introduction

Heterogeneous processing [14, 16, 20, 21] aims to exploit the coarse-grain heterogeneity of an application task by executing it on a suite of heterogeneous parallel computers. In such systems, parallel computers with diverse architectures (SIMD/MIMD/vector, etc.) are connected through a network. The basic idea is to execute a subtask on the parallel computer that is most suitable for it among those available in the network.

Previous studies in heterogeneous computing have concentrated mainly on the software aspects: determining the nature of parallelism inherent in the subtasks, finding a suitable match between a subtask and a parallel computer, and the penalties of non-suitable subtask assignments [14, 21, 34]. However, very little research has been done to study the architectural and communication aspects of such systems. Since heterogeneous systems emphasize coarse-grain meta-parallelism, they involve high volume of communication that is bursty in nature. Also, innovations in the area of host network interfaces, as done in the Nectar project [22], can enable nodes of a heterogeneous system to pump data of the order of 1 gigabyte/second between their memory and the external network. Hence, there is a need for high bandwidth and low latency communication networks to build heterogeneous platforms.

Recently, high speed communication networks like HIPPI [33], FDDI [27], and Nectar [4, 22] are being used to interconnect high speed parallel computers. These networks use optic fiber as the communication medium to transmit data between the nodes. The constituent nodes in a typical heterogeneous system can pump data into the network at a rate greater than the bandwidth provided by HIPPI (100 Mbytes/sec) and FDDI (100 Mbits/sec). Obviously, the increase in communication bandwidth has not kept pace with the increase in processing speed. The theoretically available bandwidth of an optic fiber cable is orders of magnitude greater than the bandwidth achieved by HIPPI and FDDI. This raises some interesting issues: What prevents HIPPI and FDDI from achieving a higher bandwidth? Is it possible to design *communication protocols* to achieve higher bandwidths? Can the state-of-the-art technology support such protocols? Finally, how suitable are these protocols for the communication patterns exhibited by tasks executing on heterogeneous parallel systems?

This paper explores such challenging issues and shows that existing high speed networks have high latency and limited achievable bandwidth because they employ electronic network protocols. It also shows that higher bandwidth can be achieved by employing *optical switching*. The recent developments in fiber-optics networks promise data transfer rate of the order of 1 Gbits/sec. One such development is *passive star-coupled interconnection* using *wavelength division multiplexing* [5, 7, 10]. This technology is currently being explored to provide high performance interconnection for parallel systems [6, 32]. However, it has not been studied in the context of heterogeneous parallel systems.

In this paper, we develop guidelines for building heterogeneous parallel systems with passive star-coupled interconnections. Unlike studying the performance of an interconnection network in isolation, we consider an integrated approach in deriving the best communication protocol and topology for heterogeneous parallel systems using optical interconnection. We consider the programming model, task scheduling strategies, application characteristics and network performance together. The commonly used *master-slave* computation model is used to evaluate our system. In this model, the subtasks executing on different slave computers communicate with each other through message passing via the master node. As an alternative to the slaves directly sending the results of their subtasks to the master, which is known as *direct communication*, we propose two new approaches using optical interconnections: a tree-like *hierarchical interconnection* topology and a communication scheme that employs *tunable optic filters* [7] at the communicating nodes. The passive star-coupled interconnection can support a variety of communication protocols for master-slave communication, like *ALOHA* [3] or time-division multiplexed access (*TDMA*) to the shared channel. We analyze these protocols and topologies from the standpoint of *medium access control* as well as *contention resolution*.

Various combinations of communication protocols and topologies are evaluated qualitatively, as well as quantitatively through analytical modeling and simulations. It is shown that hierarchical interconnection and tunable optical filters at the master node can reduce channel contention in the network in a significant manner.

The paper is organized as follows: In Section 2 we provide an overview of heterogeneous parallel system, and the master-slave computation model. Section 3 evaluates the processing, communication and disk I/O requirements of heterogeneous systems. In Section 4 we describe the proposed communication strategies, develop an analytical performance model for the the strategies, and demonstrate that the performance equations are intractable. Therefore, Section 6 quantitatively evaluates the performance of the proposed communication schemes through simulation. The suitability of different communication strategies for building practical heterogeneous systems is discussed in Section 7. Finally, the conclusions are presented in Section 8.

2 Heterogeneous Parallel Systems

A heterogeneous parallel system consists of a suite of autonomous parallel computers connected to each other by a high-speed communication network. Such a system is suitable for executing applications that exhibit coarse grain meta-parallelism. In this section, we provide an overview of heterogeneous parallel architectures, task graphs with precedence relationships, and a master-slave execution model to execute task graphs on heterogeneous systems.

2.1 A Typical Heterogeneous Parallel Architecture

A heterogeneous parallel system consists of a set of parallel computers, each designed to exploit a different kind of parallelism (SIMD/MIMD/vector, etc.). Each computer has its own disk. The computers can communicate with each other through a high speed communication network connecting them. An example of such a system is shown in Fig. 1.

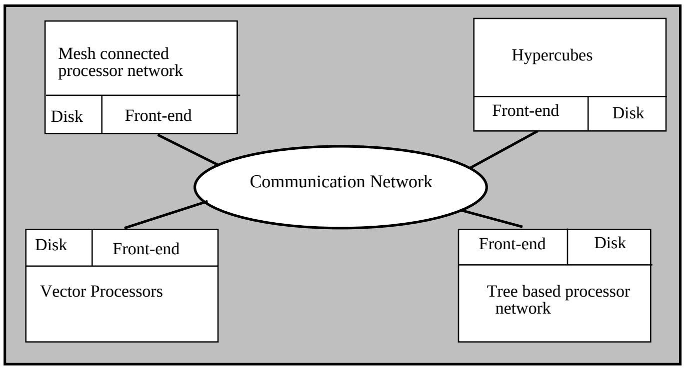

Figure 1: An example heterogeneous parallel system with diverse architectures.

Applications targeted for heterogeneous systems exhibit coarse granularity [13]. They can be divided into a number of subtasks having different kinds of parallelism embedded in them (SIMD/MIMD/vector). These subtasks also have certain control and data dependencies amongst them. Based on these dependencies, the execution of the application task can be expressed as a precedence relationship among the subtasks, as shown in Fig. 2. Two subtasks with no precedence

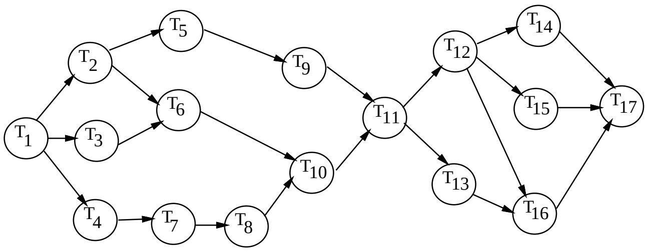

Figure 2: A coarse grained task graph showing precedence relations between subtasks.

relation between them can be executed concurrently, typically on separate parallel computers.

2.2 A Representative Master-Slave Computation Model

Tasks suitable for heterogeneous computing are large scientific computations in diverse areas like medical imaging, fluid dynamics, finite-element analysis, combinatorial analysis, protein sequencing, etc. [21, 23]. Based on the precedence relations, some of the subtasks can be executed concurrently on different computers in the network. Let there be n_1 subtasks that execute concurrently on a set of n_1 computers and produce their respective results. Let there be n_2 subtasks, mapped onto a different set of computers, that use the results produced by all these n_1 subtasks as their input data. Using a commonly used multiple-phase master-slave model as shown in Fig. 3, the master computer spawns n_1 subtasks on different slave computers in the beginning of a phase. Each slave sends its result to the master on completion. The phase ends when the master has received results

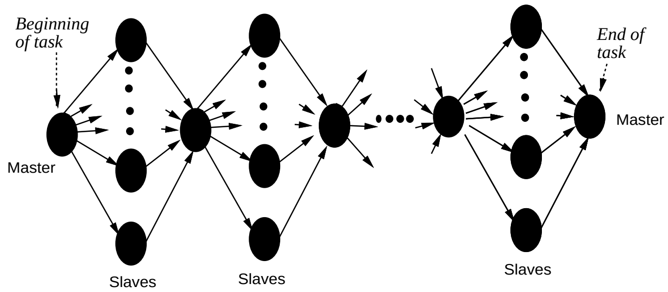

Figure 3: A multiphase master-slave model of task execution.

from all the n_1 subtasks. In the next phase the master spawns n_2 subtasks on different slaves and sends the appropriate input data to each of them. Thus, the number of messages required to distribute the results of the n_1 subtasks to the n_2 subtasks succeeding them is $n_1 + n_2$. This compares favorably with the $n_1 \times n_2$ messages required for point-to-point communication between these two sets of slaves. Such master-slave model, with centralized information about all computers, also helps in achieving load balancing in a heterogeneous system [17]. It is to be noted that the number of subtasks spawned concurrently in a phase is known as the *degree of parallelism* of that phase.

The task graph shown in Fig. 2 is quite suitable for multiple-phase master-slave execution. Initially, the master spawns T_1 on a slave. On its completion, subtasks T_2, T_3 and T_4 are spawned concurrently on three different slaves. The completion of these subtasks marks the end of the second phase. The master collects their results and then concurrently spawns T_5 and T_6 on two different slaves and schedules T_7 followed by T_8 on a third slave computer. The completion of these four subtasks marks the end of the third phase. Similarly, T_9 and T_{10} correspond to the fourth phase, and so on.

2.3 Task Scheduling Strategy

Besides the communication overheads associated with the master-slave model of execution, the completion time of a task depends on the scheduling strategy employed to assign its subtasks to the different nodes of the heterogeneous system. A good strategy is one that schedules a subtask on a machine that exploits the kind of parallelism embedded in the subtask. It also considers the capacity and the bandwidth of the main and auxiliary memory at the nodes of the system as they can influence the task completion time. In heterogeneous systems all the tasks are generally submitted at one particular node which acts as a front-end for the entire system. First, *code profiling* [21] is done to identify the different kinds of parallelism in a task. Based on this information, the task is broken down into several subtasks, each exhibiting a different kind of parallelism. Then *analytical benchmarking* [21] is done to determine the suitability of a particular computer for a given type of parallelism. Subsequently, the operating system schedules the subtasks on the different computers in the network.

Without loss of generality, let us assume that there are always a sufficient number of computers available, and each subtask can be allocated to the computer which best exploits the nature of parallelism in that subtask. This assumption is based on the *Optimal Selection Theory* [21, 34]. It can also be assumed that only one task is being executed by the system at any given time. This avoids external interference from any other task. Also, the main memory at a slave computer is assumed to be big enough to store the executable code of any subtask and the data that it works with. The code for the subtask already resides at the slave to which it is mapped [29]. So, execution of a subtask at a slave is similar to a remote procedure call [31]. The master makes the remote procedure call, and the remote procedure is said to be executed at the slave. However, unlike remote procedure calls, the master does not block until it has spawned subtasks at all the slaves in a phase. Now, let us examine the design intricacies of such a task scheduling strategy for heterogeneous parallel systems. It involves data migration between master and slaves, and *no code migration.*

No Code Migration

In this strategy, the master computer sends only the input data (not the executable code) for the subtask to the appropriate slave computer. It is assumed that on receiving the data, the slave invokes the routine corresponding to the subtask from its library. The actual implementation of the subtask at a slave is optimized to exploit the architectural features of the slave, and to provide the best possible performance for the subtask at that computer. Figure 4 represents the execution of a phase consisting of n subtasks. While the master is migrating the data for one subtask, other subtasks, belonging to the same phase, whose data was sent earlier, may be executing at the corresponding slaves. This leads to an overlap between computation and communication in execution of the application.

Figure 4 shows the execution of a phase consisting of n subtasks. The thicker ovals represent actions taken by the master. In this model *transmit* t_i indicates the transmission of the data for the i^{th} subtask to the appropriate slave. The completion time of a phase is equal to $Max_{i=1}^{n}(T_i)$, where T_i is the time at which the result from the i^{th} subtask reaches the master.

Thus, the completion time of a phase consisting of n subtasks can be expressed as:

$$Max_{i=1}^{n}[t_i^{start} + t_i^{comm} + t_i^{execute} + t_i^{paging}] \tag{1}$$

where $t_i^{execute}$ is the time to execute the i^{th} subtask at the corresponding slave, In this expression, t_i^{start} is equal to the time elapsed between the start of the phase and the instant that the master starts sending the data for subtask i to the corresponding slave. This is equal to:

$$\sum_{j=1}^{i-1}[t^{setup} + (t^{transfer} \times \mid data_j \mid)] \tag{2}$$

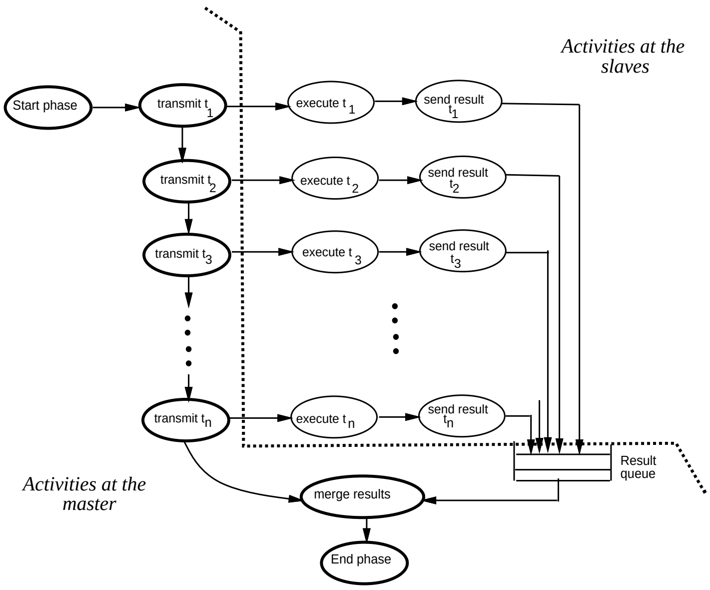

Figure 4: Temporal representation of master-slave activities in a phase (t_i: subtask i).

where t^{setup} is the communication set-up time and $t^{transfer}$ is the time to transfer one unit of data from the master to a slave. The term $\mid data_j \mid$ denotes the amount of data sent from the master to slave j. The component t_i^{comm} is the time taken to communicate the data and result for subtask i between the master and the slave, and is equal to:

$$2 \times t^{setup} \ + t^{transfer} \times \mid data_i \mid + t^{transfer'} \times \mid result_i \mid \tag{3}$$

where $t^{transfer'}$ is the time to transfer one unit of result from a slave to the master. It is to be noted that $t^{transfer}$ and $t^{transfer'}$ may not be equal. This asymmetry in achievable throughput in the forward (master to slave) and backward (slave to master) directions may be due to the different communication patterns, namely, one-to-many and many-to-one, respectively. The throughput asymmetry will be analyzed in the context of communication topology and protocols in Section 4, and quantitatively evaluated through simulation experiments in Section 6.

The component t_i^{paging} is the time spent fetching pages of the executable code for subtask i into the main memory of the slave's front-end processor, from the library in auxiliary storage. If the code size of the subtask is equal to S_i, there will be $\lceil S_i/P \rceil$ page faults, where P is the size of a page. If the time to fetch a page from auxiliary memory is equal to t^{disk}, t_i^{paging} is equal to $\lceil \frac{S_i}{P} \rceil t^{disk}$. We assume that the main memory of a slave is big enough to store the code and data of its subtask. A smaller main memory may lead to *additional page-faults*, whose number would depend on the relative size of the memory and the subtask code and data, and the pattern of reference. Considering the rate at which the size of the main memory of computers is increasing we can safely assume that the executable code, data, and result for a subtask will fit in the main memory. So, we do not consider the additional page-faults mentioned above.

3 Architectural and Communication Requirements

The processing speed of the nodes on which the subtasks are scheduled will have a significant impact on the task completion time. If the communication network between the nodes has a low bandwidth and/or high latency the task completion time will increase. As the problem size grows, greater demands are also placed on auxiliary memory like disks to store data and provide it to the processors at a fast rate and with low latency. In this section we briefly discuss these requirements.

3.1 Processing Speed

The performance of the currently available parallel computers and of those expected to be available in the next few years has been reported in [1, 2, 15, 30]. The processing speeds of the nodes that can constitute a heterogeneous system is growing at a fast pace. A 1024 node CM-5, with a peak performance of 131 gigaflops is an example of the processing power currently available. Even if the sustainable performance is assumed to be 10% of the peak, 13.1 gigaflops can be executed every second. So, it is realistic to assume that each node in a heterogeneous system will have a comparable computational rating. In the master-slave computation model, each slave will correspond to such a node.

The time required to copy the code and data from the front-end of the slave to each of the constituent processors of the slave before the computation starts, and the time to gather the result at the front-end may contribute to the total execution time at the slave. We assume that the subtask code is broadcasted to each constituent processor while a small and distinct fraction of the data is sent to individual processors. Emerging high-speed DRAMs like RamBus and RamLink can provide memory bandwidth of up to 500 megabytes/second [1]. Modern designs used in SCI-based local area multiprocessors [2, 15] are also capable of supporting a transfer rate of 1 gigabyte/second between the front-end and the computational processors at a slave computer. Hence, the communication time between the front-end and the slaves does not lead to bottlenecks.

3.2 Disk I/O Bandwidth

With the increase in processing power, greater demands are placed on the storage disks. In the *no-code-migration* strategy the code for a subtask is initially fetched from the disk at the slave. The I/O bandwidth provided by the disk storage of the currently available supercomputers is low, compared to their data processing rate. For example, the basic disk storage node and the data vault of CM-5 provide peak transfer rates of 17 megabytes/second and 25 megabytes/second, respectively. The Intel Paragon's disk provides a peak transfer rate of 100 megabytes/second [19]. Disk transfer rates can be further increased through architectural innovations such as connecting an array of disks to the high speed network mentioned above and employing interleaved disk access.

However, over the years, disk *access time* has not decreased at the same rate as the transfer rate has increased. The disk access time can be hidden in the *no-code-migration* approach in the following manner: Let the first few bytes of the message sent by the master containing the data for a subtask carry information about the library routine to be invoked at the slave. As soon as the slave gets this information it initiates a request to fetch the code from the disk. So, the disk access time gets overlapped with the data transfer time from the master to the slave. As soon as this data transfer finishes, the slave can start receiving the subtask code from the disk.

3.3 Communication Bandwidth and Latency

The nodes in a heterogeneous system require high bandwidth communication links between them. The communication issue is complicated by the distance separating the nodes. Most of the previous solutions for handling inter-node communication in parallel systems have assumed that these nodes are on different boards inside a cabinet, or are computers sitting next to each other in a room. However, in a heterogeneous system, the nodes may be as far apart from each other as different corners of a city. In our evaluations, we assume such a separation between the constituent nodes in the heterogeneous system: typically of the order of ten kilometres corresponding to a Metropolitan Area Network.

Local area networks are currently used to connect computing nodes a few kilometers apart. The twisted pair and coaxial cable communication media currently used by these networks have a limited bandwidth over distances of a few kilometers [31]. They cannot sustain the traffic produced by a heterogeneous system, which can be of the order of several gigabits per second, over the distances separating the constituent nodes. A single coaxial cable cannot carry several data channels of 1-gigabit bandwidth, even over short distances. Establishing dedicated end-to-end coaxial cable links between every pair of nodes is not economical. Besides, several repeaters may be required along the path and each repeater will contribute towards communication latency.

The high performance parallel interface (HIPPI) physical layer standard is being suggested as the backbone for connecting high performance computers [33]. HIPPI defines full duplex parallel interfaces that can run at speeds up to 200 megabytes/second [30, 33]. It has an Ethernet connection in its data path which becomes a bottleneck. The high overhead associated with Ethernet communication also limits the data transmission rate that can be sustained. Establishing and terminating a HIPPI connection also takes a long time due to round-trip delays across the Ethernet [33].

On the basis of the preceding discussion, it is obvious that the communication media and protocols currently in use are not suitable for building future metropolitan area networks of heterogeneous systems. There is a definite need for multiple high bandwidth, low latency communication channels. In the following sections, we propose solutions to this problem by using optic fiber networks and a set of new communication strategies.

4 Communication Strategies Using Optic Fiber Networks

Technological advancements in the area of optical switching, routing and data transmission [7, 12] have enabled optic fiber networks to support multiple low latency communication channels, each with a sustainable data rate of 1 gigabyte/second. In this section, we show how optic fiber cables and an entirely new set of communication protocols and topologies can be used to support high bandwidth and low latency communication requirements of heterogeneous parallel systems.

4.1 Design Components

Three major issues (shown in Fig. 5) should be considered while designing optical networks. When

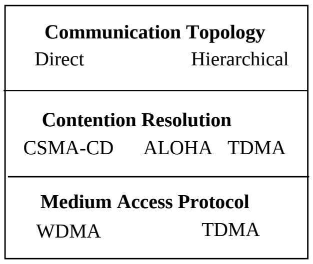

Figure 5: Design components for optic fiber networks.

multiple nodes try to use an optic fiber cable to send data, the available bandwidth can be accessed either in a time-division multiplexed (TDMA) fashion or in a wavelength-division multiplexed fashion (WDMA). Irrespective of the medium access strategy employed, there is a likelihood that multiple nodes trying to use the communication medium at the same time will lead to contention. Hence, having settled on an appropriate medium access protocol, a suitable contention resolution protocol should be adopted. The contention for the communication medium and the achievable bandwidth are also significantly influenced by the communication topology. Let us, now, discuss each of these issues in some detail.

4.2 Medium Access Protocol

The maximum rate at which data can be pumped in, or received at communication end-points in an optical network is limited by the speed of the electronic devices at these end-points. The available bandwidth of the optic fiber is four orders of magnitude larger than the peak rate of the electronic devices. Therefore, to exploit the huge bandwidth of optical networks, concurrency among multiple data streams needs to be employed [24]. Wavelength division and time division multiplexing are two ways of achieving concurrency. Recent technological advancements have made it easier to achieve higher transmission bit-rates using wavelength division multiplexing instead of time division multiplexing [7]. Also, the bandwidth of the fiber is most easily accessed in the wavelength domain than in the time domain due to the following reasons: (i) nodes need to synchronize to within one time slot in *time division multiplexed access* (TDMA), (ii) *wavelength division multiplexed access* (WDMA) employs existing technologies, and all the equipment at communication end-points can be operated at peak electronic processing speed [24]. Hence, WDMA is the most suitable medium access protocol for optical interconnection architectures.

WDMA protocols are being developed to transmit several 1-gigabit data streams through a single optic fiber cable [5, 7, 10]. These protocols assume passive optical star-coupled networks with *preassigned* channels (wavelengths). Figure 6 illustrates such a network. The individual signal streams, along different wavelengths, from the source nodes are combined in a passive star coupler. The combined signal is sent on each outgoing fiber. At the destination node optical filters can select the desired signal stream and reject the remaining streams. The protocols have low complexity, which is especially important since there is still a great mismatch between the speeds of the optical network interfaces and the bandwidth of the fiber [5]. We present brief descriptions of two WDM access mechanisms. Based on their relative merits we will select one of the access mechanisms for use in heterogeneous systems.

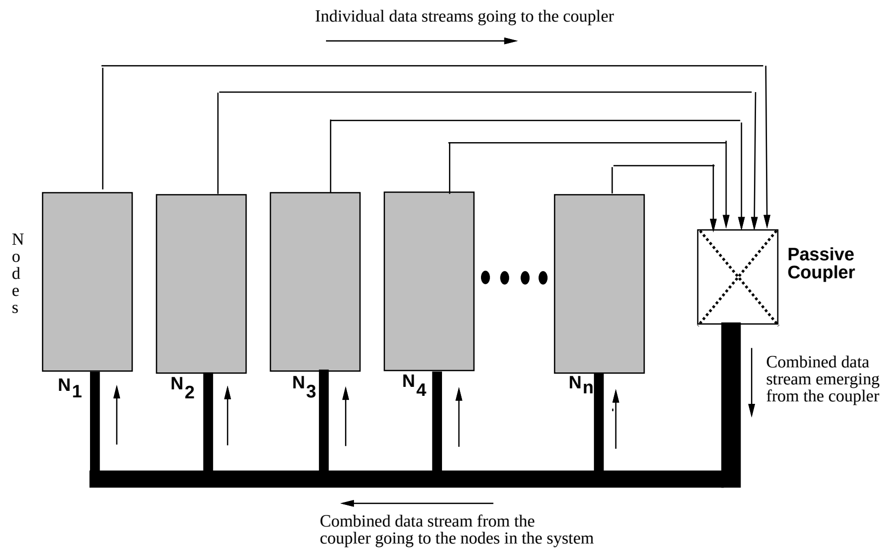

Figure 6: A passive star-coupled optical network using wavelength division multiplexing.

4.2.1 Communication with Tunable Transmitters

In this access mechanism, each node is assigned a *home frequency* (channel) at which it can receive data. The senders tune their transmitters to the receiver's channel [5]. Let us consider the impact of this access mechanism for heterogeneous parallel systems. In the beginning of a phase of the master-slave execution model, the master sends data, one at a time, to the slaves along their respective channels. Each slave attempts to send its result to the master, on the master's channel. If two or more slaves send their results to the master at the same time collision results. The impact of collisions on the performance of the system depends on the number of slaves contending for the master's channel and the duration of their transmissions.

4.2.2 Communication with Tunable Receivers

In this access mechanism, each node can transmit along only one channel – its home channel [5]. The receivers have to tune to the sender's channel to receive the data. In heterogeneous systems, once the slaves have tuned to the master's channel, the master can send data to the slaves one after the other. On finishing its computation, a slave tries to send its result to the master by requesting the master to tune to the slave's transmission frequency. If the master is tuned to some other frequency the slave waits until it gets a positive acknowledgment from the master. For master-slave communication, using tunable receivers and fixed frequency transmitters there are no collisions.

4.2.3 Qualitative Comparison

If tunable receivers are used, a separate control channel is needed to carry the tuning requests and acknowledgments between nodes. The transmitters should be able to transmit at two frequencies — their home frequency and the frequency of the control channel. Besides, tuning requests can collide on the control channel. Also, advances in the areas of fast tunable transmitters exceed those in fast tunable receivers, and the tuning times for receivers are relatively long as compared to the packet transmission times and their tuning range is small [10]. Thus a tunable receiver protocol will have significant overheads that may offset the advantages due to the absence of contention. Tunable receivers are suitable for broadcasting or multicasting data, i.e., if the input data to several slaves is the same. In most heterogeneous applications different subtasks, being executed concurrently on different computers, will be working on separate data. Consequently, the broadcasting capability of the *communication with tunable receivers* access mechanism cannot be exploited.

Hence, we choose the tunable transmitter access mechanism with the following modification: all the slaves are assigned the same home channel, and the master's transmitter is tuned to that channel. Thus, in the beginning of a phase, the master node does not have to retune to a different channel each time it finishes sending the data to one slave and has to start sending data to another slave. This can lead to significant reduction in communication time. *Beginning of message* and *end of message* markers are sent by the master to indicate when one slave's data transmission begins and ends, respectively. As we assume that only one application is executing in the system at any given time, there is no contention for the home channel of the receivers.

Having decided upon tunable transmitters for heterogeneous systems, appropriate contention resolution strategy and communication topology need to be selected.

4.3 Contention Resolution Protocol

As mentioned in Section 4.2.1, using tunable transmitters for sending results will lead to contention for the master's channel due to collisions. In order to maximize the achievable throughput in the network, such collisions have to be minimized.

4.3.1 CSMA-CD vs. ALOHA

If a carrier sense strategy, like CSMA-CD [18], is employed the slaves will send data by tuning to the master's channel, and at the same time *sense* the channel for collisions by *listening* to the channel. A slave will stop transmitting as soon as it detects a collision. Then it will wait for a random period of time before retrying to send the result. Such a strategy is not suitable for the tunable transmitter medium access protocol because the receivers at the slaves are tuned to sense signals traveling along their channel only. Tuning these receivers to sense the presence of traffic along the master's channel will introduce excessive delays.

Since the slaves' receivers are not tuned to the master's channel, a slave can know about a successful transmission to the master only on receiving an acknowledgment from the master. Hence, the CSMA-CD protocol will deliver a performance that is hardly better than the comparatively simpler ALOHA protocol [3] which does not have to continuously sense the channel. The ALOHA protocol transmits the entire data and relies on the feedback properties of the network to detect collisions. The ALOHA protocol for communication with the master during result transmission appears suitable. However, channel contention can severely curtail achievable throughput, especially if the subtask result sizes are large.

4.3.2 Performance of ALOHA Protocol

In the past, analysis of the ALOHA protocol has been mainly focused on mean throughput and delay [31]. However, mean throughput and delay numbers for the ALOHA protocol [3] are not a good indicator of performance for the master-slave model due to the following reason: the communication in the master-slave model is extremely bursty. Near the end of a phase, all the slaves may finish their execution around the same time and there will be severe contention for the master's channel. Rest of the time, there is no contention at all. So, the metric of interest to us is the achievable throughput during peak contention. The overall task completion time will be greatly influenced by this.

If the ALOHA protocol is employed, in the beginning of a phase only the master is sending data to the slaves and there is no contention. So, the achievable throughput is equal to the peak bandwidth of the channel. For the end of the phase, let us assume that there are n slaves. Let p be the probability that a slave is trying to transmit its result to the master at a given time instant near the end of a phase. It is to be noted that p is the conditional probability, where the condition corresponds to *end of a phase*. Even if the general probability of a slave trying to send its result to the master at any point of time may be low, p can be high.

Therefore, the probability that some slave successfully transmits its result to the master when all n slaves are contending for the master's channel, near the end of a phase, is equal to:

$$P = n \times p \times (1-p)^{n-1} \tag{4}$$

A high value of P would indicate fewer collisions, and higher achievable throughput. To find the optimal value of p (which maximizes P), the following expression needs to be solved for p:

$$\frac{dP}{dp} = 0 \quad \Rightarrow \quad \frac{d(n \times p \times (1-p)^{n-1})}{dp} = 0 \tag{5}$$

On solving this expression, the best value of p is $1/n$. Substituting $p = 1/n$ in P, we get:

$$P_{optimal} \quad = \quad (\frac{n-1}{n})^{n-1} \tag{6}$$

When the number of slaves contending for the master's channel is small, the probability of a slave successfully transmitting its result, in the face of contention from other slaves, is high. However, this probability diminishes quite fast as the number of slaves increases. When the number of slaves is as low as 5, the probability drops very close to its asymptotic value of $1/e$. This provides two insights into the achievable throughput during result transmission from slaves to master:

1. The time needed by a slave to send its result to the master, in a no contention situation, increases with the size of the result. So, near the end of a phase, p increases with increase in result size. For optimal performance p should be equal to $1/n$. Hence, with fixed n, as the size of the result increases, the achievable throughput increases until $p = 1/n$. Beyond that, the achievable throughput decreases with increase in the size of the result. This should get reflected as a sharp increase in the overall task completion time, with increase in the size of input/result.

2. Keeping the result size fixed (i.e., fixed p), the achievable throughput increases with increasing degree of parallelism until $n = 1/p$. Beyond that, increasing degree of parallelism leads to increased contention, and reduced achievable throughput. Hence, initially, the task completion time should decrease with increasing degree of parallelism, as the inherent parallelism is being exploited and there is low contention for the master's channel. Beyond a certain point, the increase in channel contention should offset the impact of parallelism, and the task completion time should increase with increasing degree of parallelism.

As contention for the master's slave is a critical factor in determining the achievable throughput, and therefore the task completion time, contention reduction is an important issue. One approach to contention reduction is to consider reservation based channel access schemes.

4.3.3 Reducing Contention with TDMA

A suitable, yet simple, reservation based scheme is time-division multiplexed access (TDMA) to the master's channel [31]. In this strategy, an interval of time, called a TDMA cycle time, is divided into slots of finite duration. Each slave is assigned a time slot and can directly send its result to the master, along the master's channel, only during this slot. If the slave misses its time slot, it has to wait for its slot in the next cycle. If the number of slaves in a phase is less than or equal to the number of slots in a cycle, each slave gets its own slot and there is no contention among the slaves. This time-division multiplexed access to a channel is distinct from the TDMA medium access protocol shown in Figure 5.

However, TDMA lacks flexibility because the number of slots in a cycle and the duration of each slot are fixed *a priori*. If the number of slots is more than the number of slaves in a phase, several time slots remain *unutilized*. Fewer slots than the number of slaves in a phase will lead to *contention* among the slaves for some slots. If the slot duration is greater than the time needed by a slave to send its result to the master, the slot is *underutilized*. So, TDMA running on top of the WDM medium access protocol will have a good performance only for a small subset of applications whose degree of parallelism and slave result sizes match well with the number of slots in a cycle and the slot duration, respectively.

Therefore, the ALOHA protocol and the TDMA protocol appear to be suitable for use with tunable transmitters and will be quantitatively evaluated through analytical modeling and simulation experiments.

4.4 Communication Topology

The ability of a contention resolution strategy to minimize collisions depends in great part on the logical communication topology and the number of nodes contending for a given communication channel in a phase (degree of parallelism). Let us evaluate communication topologies which can deliver better performance for communication with tunable transmitters.

4.4.1 Direct Communication Topology

In the master-slave model, this communication topology implies that all the slaves directly send their results to the master node, at the end of a phase, on the master's channel by tuning their transmitters to that channel. This is the simplest topology and can be blended well with the ALOHA protocol. Each slave will send its result along the channel without checking the status of the channel. There may be collisions between the data streams. So, corresponding slaves will have to back-off and retry after a random period of time. As the degree of parallelism of a task increases, more slaves in a phase are concurrently trying to send their results to the master. Hence, the chances of collision rise. If this problem is not addressed, the expected decrease in task completion time, due to increased parallelism, will be offset by the contention penalty. If a single communication channel is being shared by all the slaves, there will be great contention for the channel. The contention will also increase with increase in the size of the result that the slaves have to send to the master.

4.4.2 Limitations of Direct Communication Topology

Given that the size of slave results, and consequently p, is fixed, optimal throughput can be achieved if the degree of parallelism (n) is equal to $1/p$, as discussed in Section 4.3. If the degree of parallelism inherent in the task being executed is less than $1/p$, increasing the throughput is beyond the control of a designer of a heterogeneous parallel system. However, if the inherent parallelism in the task is greater than $1/p$, it is desirable to somehow bring down the number of slaves contending for a channel to $1/p$, while at the same time exploiting all the inherent parallelism in the task.

For tasks exhibiting high degrees of parallelism, direct communication topology has two limitations:

1. If the number of contending slaves is greater than $1/p$ there is high contention and optimal throughput is not achieved.

2. High degrees of parallelism require a large passive star coupler. However, lack of scalability of passive star couplers is a severe limitation.

Thus a communication topology capable of delivering good performance over a wide range of degrees of parallelism in an application is needed. We propose a new *hierarchical communication topology* to deliver better performance as it reduces the impact of contention, and also alleviates the problem of scalability.

4.4.3 Hierarchical Communication Topology

In this topology, additional nodes, called sub-masters, are introduced into the system. The slaves in a phase are assigned to mutually disjoint sets with each set corresponding to a sub-master. The slaves in a set send their results to their sub-master. There is a passive star coupler for each sub-master and its half of the slaves. Once a sub-master gets the results of all its slaves, it forwards them to the master. Thus, each slave has to contend for the communication channel leading to the sub-master with only those slaves that belong to its set. In this hierarchical communication topology, the number of sub-masters contending with each other for a channel, to send the results to the master, is significantly less than that in the direct communication topology.

Such hierarchical communication topology, with tunable transmitters, using WDM for use in heterogeneous systems is shown in Fig. 7. The master directly sends the input data for all the subtasks to the corresponding slaves without the involvement of the sub-masters. At the end of their computations, half the slaves send their results to one sub-master and the other half to the other sub-master. The assignment of slaves to sub-masters is done *a priori*. Once a sub-master gets the results corresponding to all its slaves, it combines and sends them to the master. The phase ends when the master has received results from both the sub-masters. Only half the slaves contend for the channel of each sub-master, and only two sub-masters contend for the master's channel. In the forward direction (master to slaves) communication of data is in a single hop, while communication of results in the backward direction is in two hops. Thus, the hierarchical topology is a hybrid of the single-hop and multi-hop passive star-coupled optical networks described in [24, 25].

Both TDMA and ALOHA protocols can be used in conjunction with the hierarchical topology. The slaves can have time multiplexed access to the channels of their sub-masters or they can employ the ALOHA protocol. Each sub-master can send the results it collects from its slaves to the master using the ALOHA protocol. In Fig. 7, the channels for which contention takes place are represented like buses for simplicity.

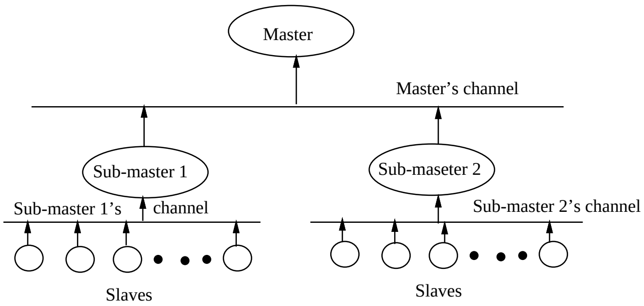

Figure 7: A hierarchical representation of channel contention during the result-receive period using two additional sub-masters.

The hierarchical topology promises a much better performance than the direct communication topology for tasks exhibiting high degrees of coarse grain parallelism because each sub-master can be assigned close to an optimal number $(1/p)$ of slaves. Hence, the overall task completion time will continue to diminish with increasing degree of parallelism, well beyond the point at which the direct topology has its best performance. However, ultimately, the task completion time will start rising with increasing parallelism even for the hierarchical topology. This is because the number of slaves assigned to each sub-master will grow beyond $1/p$, and the contention for the home channels of the sub-masters will be high.

For very large tasks with many subtasks, large size of data and result, and high degrees of parallelism, additional levels in the hierarchy between the master and the slaves may be needed to operate in the neighborhood of optimal values of n and p. The communication topology may look like a tree with the number of children of interior node increasing as one goes farther away from the master. Such a topology may require a large number of additional nodes to act as sub-masters.

4.4.4 Direct Communication with Tunable Optic Filters at Master Nodes

Using tunable optic filters is an elegant alternative to hierarchical topology. A tunable optic filter can be tuned to receive signals along one or more channels. Acoustooptic tunable filters can be tuned to receive signals on the desired channel by changing the acoustic drive frequency [7]. They have a broad tuning range and the drive power to tune such filters is quite low, 100–200 milliwatt [9]. Also, acoustooptic filters can select more than one wavelength *simultaneously* by injecting more than one RF drive signal into the acoustic transducer. Such a tunable filter, capable of selecting five wavelengths simultaneously, has been demonstrated [8]. This enables a node to receive data along multiple channels *concurrently.*

Let us consider a communication topology where the master node is equipped with an acoustooptic filter that enables it to receive data from the slaves along *two* home frequencies. Half the slaves send their results to the master along one home frequency, and the other half along the other home frequency, using the ALOHA protocol.

Thus the sub-master nodes are no longer needed. The contention between the sub-masters for the master's channel, present in the hierarchical topology, is completely eliminated by using such acoustooptic filters. Therefore, if the direct communication topology is used along with tunable acoustooptic filters, it is expected to have a better performance than the hierarchical topology. The relative advantage will be small for applications with smaller result sizes and lower degrees of parallelism. This is because the inter-submaster contention will be small in such situations. However, using the acoustooptic filters should be much more advantageous as the degree of parallelism increases and the result sizes increase.

4.5 Summary of Suitable Interconnection Strategies

Based on the above discussion, five different combinations of communication topology and contention resolution protocol appear to be interesting, and worth evaluating, for passive star-coupled optical interconnections that employ tunable transmitters. These combinations/strategies are:

1. *Direct, with ALOHA*: during each phase of execution, the slaves send their results directly to the master node on the master's channel using ALOHA protocol.

2. *Direct, with TDMA*: the slaves send their results directly to the master node employing time division multiplexed access to the master's channel.

3. *Direct, with 2 channel tunable filter*: the master is equipped with a tunable filter enabling it to simultaneously receive data along two *home channels*. Each channel is used by half of the slaves to send their results directly to the master. The ALOHA protocol is used for both the channels.

4. *Hierarchical, with ALOHA+ALOHA*: half the slaves send their results along one sub-master's channel, while the other half send their results to another sub-master. ALOHA protocol is used by the slaves. Both the sub-masters send the collected results to the master, along the master's channel, using ALOHA protocol.

5. *Hierarchical, ALOHA+TDMA*: similar to the previous combination except that the slaves used TDMA protocol, instead of ALOHA protocol, to send their results to the sub-masters.

5 Performance Modeling of Interconnection Strategies

In this section we analytically evaluate the performance of the various communication strategies, and their impact on task completion time. As channel contention is the primary concern of non-reservation based protocols, like ALOHA, we develop a probabilistic model of the contention situation. The contention characteristics thus derived are used to obtain an expression for task completion time. Task completion time is expressed in terms of a set of system parameters, and application parameters. Keeping the system parameters fixed, we try to determine the application parameters that will lead to optimal performance.

5.1 Contention Characteristics at Phase-End

Let us consider a situation where n slaves are concurrently executing different subtasks in a phase. If direct communication topology with ALOHA is assumed, near the end of the phase each slave is trying to send its result to the master with probability p. A slave succeeds in sending its result if it is the only node sending the result, and none of the other $n-1$ slaves is sending the result. If multiple slaves try to send their results at the same time, we have collisions. The slaves whose results collide have to attempt result transfer again, until they succeed. Hence, it is important to determine the expected number of attempts a slave will have to make to successfully send its result to its master. A large number of attempts before success would lead to high task completion time.

The probability that a slave succeeds in its very first attempt to send its result $= p(1-p)^{n-1}$. Let us call this probability p'. The probability that a slave succeeds in sending its result in its second attempt $= (1-p')p'$. Hence, the probability that a slave succeeds in sending its result in the k^{th} attempt $= (1-p')^{k-1}p'$.

Therefore, the expected number of attempts a slave has to make in order to successfully transmit its result to the master, E, is given by the following expression:

$$\sum_{k=1}^{\infty} k(1-p')^{k-1}p' \quad = \quad p' \sum_{k=1}^{\infty} k(1-p')^{k-1} \tag{7}$$

Let $(1-p') = y$. Then:

$$\begin{aligned}
E &= (1-y)(1+2y+3y^2+4y^3+\ldots) \qquad (8)\\
\Rightarrow yE &= (1-y)(y+2y^2+3y^3+\ldots)\\
\Rightarrow (1-y)E &= (1-y)(1+y+y^2+y^3+\ldots)\\
\Rightarrow y(1-y)E &= (1-y)(y+y^2+y^3+\ldots)\\
\Rightarrow (1-y)^2E &= (1-y)\\
\Rightarrow E &= \frac{1}{1-y}\\
\Rightarrow E &= \frac{1}{p(1-p)^{n-1}} \qquad (9)
\end{aligned}$$

Now, let us analyze the performance difference between the hierarchical and direct communication topologies, with ALOHA protocol. Let there be n slaves spawned in a phase. Then, for the direct communication topology, the expected number of attempts a slave has to make to successfully transmit its result to the master is equal to:

$$\frac{1}{p(1-p)^{n-1}}$$

In the hierarchical communication topology, the slaves are divided equally between the two sub-masters. Hence, the number of slaves contending for a sub-master's channel is equal to $n/2$. Therefore, the expected number of attempts a slave has to make to successfully transmit its result to its sub-masters is equal to:

$$\frac{1}{p(1-p)^{n/2-1}}$$

Then, the two sub-masters will be contending amongst themselves to transmit the results they have received, from their slaves, to the master. Let the probability that a sub-master is trying to send its result to the master, near the end of a phase, be equal to $\tilde{p}$. Then, the expected number of attempts a sub-master has to make to successfully transmit its result to the master is equal to:

$$\frac{1}{\tilde{p}(1-\tilde{p})}$$

From the analysis above, it is apparent that for the same degree of parallelism (number of slaves in a phase), a slave has to make fewer attempts to successfully transmit its result to the master for the hierarchical topology than for the direct communication topology. The analysis, though simplistic, gives a fair idea of the difference in performance between the two communication topologies.

The advantage of the hierarchical topology over the direct communication topology will be illustrated by the following example. Let $p = 0.1$, and the number of slaves in a phase (n) be equal to 20. For the direct communication topology, the expected number of attempts for a slave to send its result to the master is equal to $\frac{1}{0.1\times(0.9)^{19}} = 74.03$. In the hierarchical topology, ten slaves will be contending for each sub-master's channel. Hence, the expected number of attempts for a slave to send its result to its sub-master is equal to $\frac{1}{0.1\times(0.9)^{9}} = 25.81$. So, the hierarchical topology is scalable.

5.2 Task Completion Time

The completion time of a task depends on the following factors: degree of parallelism inherent in the task, computational complexity of the subtasks, computational rating of the processors, data and result sizes, object code size, disk I/O bandwidth, communication bandwidth, and communication topology and protocol. Let us parametrically compute the task completion time.

Let the number of subtasks in a task be S, and the mean number of slaves spawned in each phase of execution be n. Then, the expected number of phases in the task's execution is equal to S/n. Let each subtask's data and result be of the same size, and correspond to $N \times N$ matrices, with the computational complexity of each subtask being $O(N^2)$ or $O(N^3)$. Without loss of generality, let us assume that the computational complexity of each subtask in a phase is $O(N^3)$. Let the computational rating of the slaves be C operations/second, and the communication bandwidth of the channels be B bits/second. Then, the time to execute a subtask on a slave is equal to aN^3/C, where a is a constant. The elapsed time in a phase, when the data for the n^{th} subtask of the phase has been fully transmitted to the corresponding slave is equal to $n(t+bN^2/B)$ where t is the communication start-up time and b is a constant representing the number of bits in each matrix element. Once all the data has been received by the slave, time taken by it to fetch the subtask's code from its disk is equal to $\mid code \mid /D$, where D is the disk I/O bandwidth.

Direct with ALOHA: For direct communication topology, the expected time for a slave to send its result to the master is equal to $(t+bN^2/B)\times E$, where E is the expected number of attempts for a successful transmission. Substituting the value of E derived above, the expected result transmission time is equal to $(t+bN^2/B)\times\frac{1}{p(1-p)^{n-1}}$, where p is the probability of a slave trying to send its result to the master at a given time instant. As p is a function of the result size, the expected result transmission time can be expressed as $(t+bN^2/B)\times\frac{1}{\alpha N^2(1-\alpha N^2)^{n-1}}$, where α is a constant value.

Hence, the expected completion time of a phase is equal to the following expression:

$$n(t+\frac{bN^2}{B})+\frac{\mid code \mid}{D}+\frac{aN^3}{C}+(t+\frac{bN^2}{B})\frac{1}{\alpha N^2(1-\alpha N^2)^{n-1}} \tag{10}$$

As code size, communication bandwidth, computational rating of the slaves, and disk I/O bandwidth are assumed to be constant, the expression can be simplified to the following:

$$(n+\frac{1}{\alpha N^2(1-\alpha N^2)^{n-1}})(t+a_1N^2)+a_2N^3+a_3 \tag{11}$$

where a_1, a_2, and a_3 are constants.

Therefore, the task completion time for direct communication with ALOHA is equal to:

$$T_{DA} = \frac{S}{n}[(n+\frac{1}{\alpha N^2(1-\alpha N^2)^{n-1}})(t+a_1N^2)+a_2N^3+a_3] \tag{12}$$

Hierarchical-ALOHA+ALOHA: For the hierarchical communication topology with ALOHA, the data transmission time, subtask execution time, and time to fetch code from disk are the same as mentioned above. The only difference is in the time to send results of slaves to the sub-masters, and then from the sub-masters to the master. If there are n subtasks in a phase, only $n/2$ of them contend for each sub-master's channel. Hence, the task completion time for the hierarchical communication topology, T_{HAA}, is equal to:

$$\frac{S}{n}((n+\frac{1}{\alpha N^2(1-\alpha N^2)^{\frac{n}{2}-1}})(t+a_1N^2)+\frac{t+a_2N^2}{\beta N^2(1-\beta N^2)}+a_3N^3+a_4) \tag{13}$$

where α, β, a_1, a_2, a_3, and a_4 are constants. The component $(t+a_2N^2)/(\beta N^2(1-\beta N^2))$ corresponds to the time for sub-masters to communicate the result to the master.

Direct-2 channel tunable filter: When 2-channel tunable optical filters are used at the master, the expected phase completion time is equal to that of the hierarchical-ALOHA+ALOHA strategy minus the expected time for the sub-masters to send their received results to the master. Thus, this time is equal to:

$$\frac{S}{n}((n + \frac{1}{\alpha N^2(1-\alpha N^2)^{\frac{n}{2}-1}})(t + a_1 N^2) + a_3 N^3 + a_4) \tag{14}$$

Direct with TDMA: When the TDMA protocol is used by the slaves to send their results to the master, there is no contention if the number of slots in the TDMA cycle is greater than the number of slaves in a phase. Let there be s slots in a TDMA cycle, each of duration $\tilde{t}$. When a slave is ready to send its result to the master, the expected time it has to wait before its slot starts is equal to $\frac{s \times \tilde{t}}{2}$. Once the transmission of the result begins, the time taken to finish the transmission is equal to $\lceil \frac{bN^2}{B\tilde{t}} \rceil \times s \times \tilde{t}$ seconds. Hence, task completion time is equal to:

$$\frac{S}{n}[n(t + a_1 N^2) + a_2 N^3 + (\lceil \frac{bN^2}{B\tilde{t}} \rceil + \frac{1}{2})s\tilde{t} + a_3] \tag{15}$$

Hierarchical-ALOHA+TDMA: As the TDMA protocol is used by the slaves, we can assume that there are $s/2$ slots per cycle, as opposed to s slots when all the slaves sent their results directly to the master. Hence, the expected waiting time between the instant a slave is ready to send its result to the sub-master and the beginning of its first transmission slot is equal to $\frac{s \times \tilde{t}}{4}$. Once the transmission of the result begins, the time taken to finish the transmission is equal to $\lceil \frac{bN^2}{B\tilde{t}} \rceil \times s/2 \times \tilde{t}$ seconds. The contribution due to the ALOHA protocol by the sub-masters, is equal to $(t + a_2 N^2)/(\beta N^2(1 - \beta N^2))$. Hence, task completion time is equal to:

$$\frac{S}{n}[n(t + a_1 N^2) + a_2 N^3 + (\frac{1}{2}\lceil \frac{bN^2}{B\tilde{t}} \rceil + \frac{1}{4})s\tilde{t} + \frac{t + a_2 N^2}{\beta N^2(1-\beta N^2)} + a_3] \tag{16}$$

From the expressions derived above, it is apparent that the task completion time for the Direct, with TDMA strategy will progressively decline as the degree of parallelism increases. This is under the assumption that the number of slots per TDMA cycle is greater than the number of nodes trying to send their results in a cycle. For all the other strategies, we expect the task completion time to decline with increasing parallelism, until a threshold is reached. Beyond that threshold, the contention between the slaves for the shared channel will more than offset the performance gains due to increased parallelism. This will lead to an increase in task completion time beyond the threshold degree of parallelism. However, the threshold point is not obvious from the expression above. Hence, we evaluate the threshold degree of parallelism next.

5.3 Solution for Minimum Task Completion Time

The values that are of interest are T_{DA} and T_{HAA} as they involve probabilistic analysis to account for collisions. In order to compute the desirable degree of parallelism (n) for minimum task completion time, given the data size (N), we need to differentiate the task completion time with respect to n, and solve for the differential being equal to zero.

To find the degree of parallelism leading to minimum task completion time for the direct with ALOHA approach, Equation (12) is differentiated with respect to n, and solved for value of n when the differential is equal to 0. The value(s) of n so obtained should then be substituted in the second differential of T_{DA} with respect to n. If the result is positive, the substituted value of n corresponds to a minima, otherwise, it corresponds to a maxima.

$$\frac{dT_{DA}}{dn} = 0$$

This implies that:

$$\frac{-S(a_3 + a_2 N^3 + (n + \frac{(1-\alpha N^2)^{1-n}}{\alpha N^2})C)}{n^2} + \frac{SC}{n}(1 - \frac{(1-\alpha N^2)^{1-n}\log(1-\alpha N^2)}{\alpha N^2}) = 0 \quad (17)$$

The equation appears to involve transcendental functions of the variables in an essentially non-algebraic way [35]. On solving the equation, n can be expressed as follows:

$$n = \frac{2S(a_3+a_2N^3+(n+\frac{(1-\alpha N^2)^{1-n}}{\alpha N^2})(a_1N^2+t))}{n^3} + \frac{(1-\alpha N^2)^{1-n}S(a_1N^2+t)\log(1-\alpha N^2)^2}{\alpha n N^2}$$
$$- \frac{2S(a_1N^2+t)(1-\frac{(1-\alpha N^2)^{1-n}\log(1-\alpha N^2)}{\alpha N^2})}{n^2}$$

It is to be observed that it is not possible to obtain the value of n for a local minima exclusively in terms of the other system parameters. Similarly, $dT_{HAA}/dn = 0$ cannot be solved for n exclusively in terms of other system parameters. Therefore, a solution cannot be obtained analytically. This implies that it is not possible to analytically compare the relative performance of the five topology-protocol combinations. However, such a comparison is necessary to determine which of these five strategies is most suitable.

There may be situations when one strategy is better than the rest. In other situations the same strategy may be inferior than other strategies. The inability to solve for values of n that lead to optimal performance implies that the cross-over points for performance cannot be determined analytically. So, in the following section we evaluate the various communication topology-protocol combinations through simulation experiments.

6 Simulation Experiments and Results

The proposed communication strategies are designed to reduce communication latency and increase the communication bandwidth for heterogeneous parallel systems. Improvements in communication characteristics of the system should be manifested as a decrease in the overall task completion time. As performance cannot be analytically determined, as explained in Section 5, in order to observe the impact of the proposed strategies, their performance was studied through simulation experiments. A process-oriented discrete-event simulator using CSIM [28] was used.

6.1 Simulation Environment

The salient features of the simulation environment are described below:

System characteristics: We simulated the performance of a heterogeneous system consisting of a set of parallel computers (nodes). Each node in the system was assumed to be have computational power equivalent to a 1024 node CM-5. The sustainable computational rating of each slave node was equivalent to 13.1 gigaflops (10% of the peak performance of a 1024 node CM-5). A sustainable data transfer rate of 100 megabytes/second between the local disk and the main memory of a node was assumed.

Communication characteristics: We assumed that the nodes were connected by a passive star-coupled network with fixed frequency receivers and tunable transmitters. The home channel corresponding to each node was assumed to have a sustainable bandwidth of 1 gigabyte/second. The mean distance between a master and a slave was assumed to be about 10 kilometers. The speed of light in a medium being inversely proportional to its refractive index, the speed of light in the fiber is 2×10^8 meters/second, giving a mean message propagation delay of 50 microseconds. We added to that about 10 microseconds as the message set-up time. Hence, a total of 60 microsecond delay was assumed between the time the first bit of a message is sent by the transmitting node and the time instant the corresponding bit arrives at the destination node. For TDMA communication, we assumed each time slot to be of 1 millisecond duration which is the time required to transmit a 400×400 matrix over the 1 gigabyte/second links we simulated. When tunable optic filters at the master were simulated, we assumed that the filter could be tuned to simultaneously receive data along two different home channels. Each home channel of the master was assigned to half of the slaves in a given phase *a priori*.

Application characteristics: The master-slave computation model was considered. Each task consisted of 1000 subtasks executed in multiple phases, as shown in Fig. 3. Both the input data and result of a subtask were assumed to be dense matrices of equal size, with normal distribution. If the mean number of rows/columns of the matrices was N, then the standard deviation was assumed to be $0.3N$. Two sets of simulations were done:

- The first set studied the impact of data size on channel contention and overall task completion time. It was assumed that the number of subtasks executed concurrently in a phase was uniformly distributed in the range 7–13 giving an average of 10 subtasks per phase. For this set of experiments using TDMA protocol, the TDMA cycle was divided into 13 slots, each of 1 millisecond duration.

- The second set studied the impact of degree of parallelism on channel contention and overall task completion time. The data and result size was kept fixed while the number of subtasks executing concurrently in a phase was varied. For these simulations, the number of slots in a TDMA cycle was not fixed at 13. Instead, it was set to be equal to the maximum number of slaves spawned in any phase of that run. So, for example if the number of slaves in a phase was uniformly distributed between 22 and 28, with the mean number of slaves per phase equal to 25, the TDMA cycle had 28 time slots of 1 millisecond each. This ensured that there was no contention between slaves for a time slot.

Each subtask in our model problem consisted of matrix operations. We used matrix operations just as a tool for modeling purposes. Subtasks may consist of a variety of different operations, depending on the nature of parallelism embedded in them. We assumed the subtask executable code size to be normally distributed with a mean of 8 megabytes and a standard deviation of 0.8 megabytes – a realistic assumption because usually the object code for a subtask has to be linked with several software libraries that make a significant contribution to the size of the executable code. Before a subtask begins execution at a slave, the executable code is fetched from the local disk to the main memory at the rate of 100 megabytes/second.

A significant fraction of numerically-intensive computations using $N \times N$ matrices involve LU-decomposition, finding the inverse, multiplication, or some iterative operations (for convergence to a result). The computation complexities of such operations are between $O(N^2)$ and $O(N^3)$. Hence, we modeled the time complexity of a subtask to be either $c_1 N^3$ or $c_2 N^2$, where c_1 and c_2 are integers, selected from a uniform distribution with ranges 5–10 and 20–100, respectively. The computational complexity of each subtask (N^2 or N^3) was also selected randomly.

Properties studied: We simulated the performance of the five communication topology–protocol combinations listed at the end of Section 4. For these different topology–protocol combinations, we studied the following effects on channel contention and task completion time:

1. effect of varying data size
2. effect of varying degree of parallelism

To study the impact of data size, different matrix sizes were simulated by varying N from 50 to 1200, with each element of the matrix being a 64-bit floating-point number. The impact of the degree of parallelism on task completion time was studied by keeping the data size fixed ($N = 800$) and varying the average number of slaves in a phase between $5 - 40$.

6.2 Effect of Varying Data Size

The relationship between subtask data size and task completion time was evaluated for the five communication topology-protocol combinations mentioned in Section 4. The simulation results obtained are shown in Fig. 8.

When the size of the subtask results (assumed to be of the same size as the subtasks' input data) is small the communication time is small. Hence, the collision probability for the ALOHA protocol is low and channel contention does not have a significant impact on the task completion time. For small result sizes ($N \leq 900$) all communication strategies have similar impacts on task

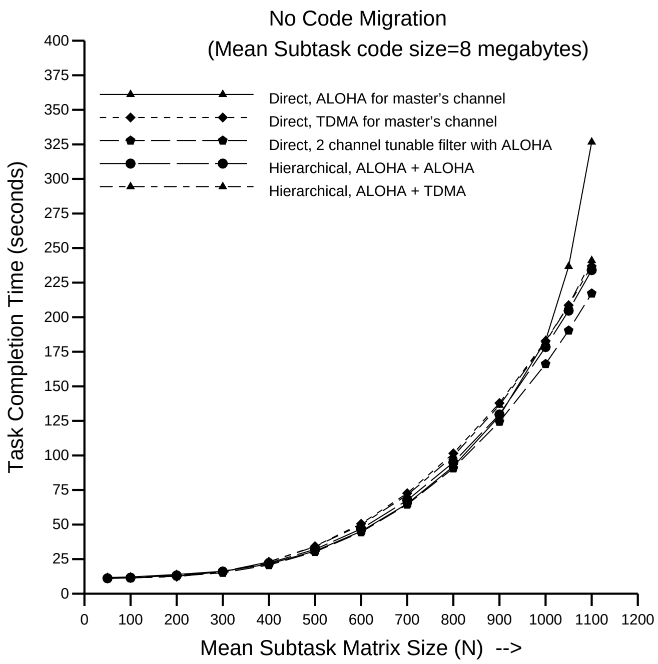

Figure 8: Effect of channel contention with varying data size on task completion time. Number of slaves per phase = 7–13 (mean = 10).

completion time. But for larger result sizes ($N \geq 900$) the ALOHA protocol has many more collisions. Consequently, the task completion time becomes higher. It should be noted that the result size (the number of elements in the matrix) is proportional to N^2. Hence, a sharp rise in the task completion time is observed for higher values of N. This is consistent with the behaviour predicted in Section 4.3.

Direct-TDMA and hierarchical-ALOHA+TDMA strategies have almost identical performance and they are both better than the direct-ALOHA strategy for large values of N. This improvement in performance is due to the reduction/elimination of channel contention. As mentioned earlier, for the direct-TDMA protocol, the maximum number of slaves in a phase was 13 and the number of slots in each cycle was kept at 13. So, there is no contention between slaves for the same time slot. However, with the number of slaves in a phase always being less than or equal to 13, some slots are not utilized. Also, some slots are underutilized if there is not enough data to fill the slots. Therefore for intermediate data sizes ($400 \leq N \leq 900$) the direct-TDMA strategy has a slightly higher task completion time than the direct-ALOHA strategy.

The hierarchical-ALOHA+TDMA strategy eliminates contention between the slaves for the sub-master's channel. Also, only two sub-masters contend for the master's channel. Thus channel contention is significantly reduced. For smaller result sizes ($N \leq 900$) channel contention is low to begin with. So, the reduction in contention is not significant. However, the results of subtasks have to be transmitted twice: once from the slaves to the sub-masters, and then from the sub-masters to the master. Hence, the hierarchical-ALOHA+TDMA strategy has a slightly higher task completion time than the direct-ALOHA strategy for intermediate data sizes ($400 \leq N \leq 900$). But, for large data sizes the elimination of contention between slaves for each sub-master's channel more than offsets the cost of transmitting the results twice.

The hierarchical-ALOHA+ALOHA strategy has a slightly better performance than the three strategies mentioned above. This is due to the fact that only half the slaves in a phase are contending for a sub-master's channel and only two sub-masters are contending for the master's channel. This is consistent with the analysis presented in Section 5. Also, this strategy does not suffer from the non-utilization or underutilization of time slots seen in direct-TDMA and hierarchical-ALOHA+TDMA strategies.

The direct communication strategy using two channel tunable filter at the master has the best performance. This is due to three reasons. First, due to the availability of two home channels for the master the contention between the slaves is reduced just as in the hierarchical-ALOHA+ALOHA strategy. Second, the contention between the sub-masters for the master's channel, seen in the hierarchical strategies, is absent because the slaves send their results directly to the master. Third, due to direct communication of the results from the slaves to the masters the results do not have to be transmitted twice.

6.3 Effect of Varying Degree of Parallelism

The results presented above assumed that the degree of parallelism (number of slaves spawned in each phase) varies uniformly from 7 to 13, i.e., the mean number of slaves in every phase is 10. There are a sizable number of applications that exhibit a higher degree of parallelism. If the system has an adequate number of suitable computers to run these mutually independent subtasks concurrently, we expect the task completion time to be reduced. But, with increased parallelism there is a greater demand on the communication network at the end of each phase, when all the slaves try to send their results to the master.

The relationship between degree of parallelism and task completion time was evaluated for the five communication topology-protocol combinations. The subtask data/result size was fixed ($N = 800$). Executions of a task with 1000 subtasks were simulated with each run having a different value for the mean number of slaves in a phase, varying from 5 to 40. The simulation results obtained are shown in Fig. 9.

Keeping the mean subtask data size (and consequently, the mean subtask execution time) fixed, an increase in the degree of parallelism initially leads to a decrease in the task completion time for all the five communication strategies. Subsequent increases in parallelism lead to an increase in the completion time for all the communication strategies except the direct-TDMA protocol which continues to show a decrease in task completion time. This increase in task completion time with increasing parallelism is because of increased contention for communication channels at the end of a phase. This is because the number of slaves contending for a channel (n) has exceeded the optimal

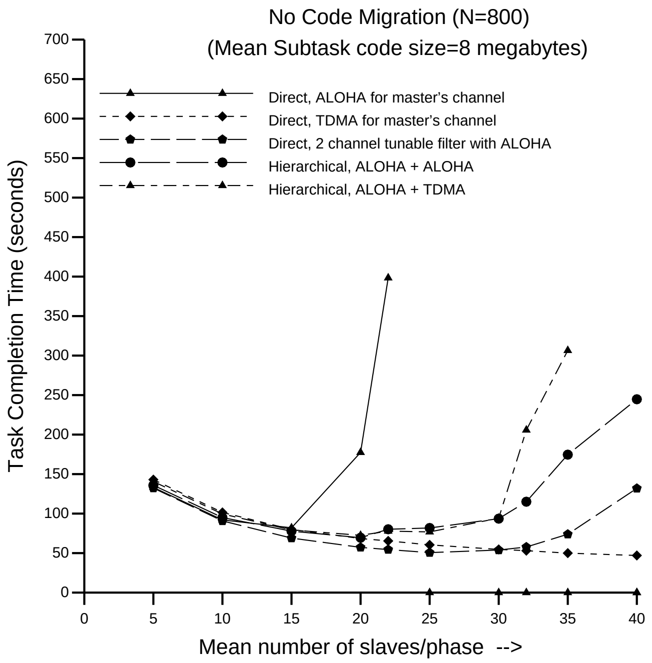

Figure 9: Effect of degree of parallelism on task completion time. Mean data size: N = 800.

value of $1/p$, where p is the probability of a slave trying to send the result on that channel at a given time instant near the end of a phase. The reduction in time due to increased concurrency is more than offset by the increase in contention.

Channel contention with increase in parallelism becomes so severe for the direct-ALOHA strategy that the task completion time increases at a very fast rate. So, its value is not plotted beyond the 20 slaves per phase point.

The hierarchical communication strategy reduces channel contention significantly which is reflected in lower task completion times. In the hierarchical-ALOHA+TDMA strategy some slots may remain unutilized in each cycle. Also, once a slave is ready to send its results to its sub-master it may have to wait for its slot. As the degree of parallelism increases, the number of slots in a cycle increases, thus increasing the waiting time. The hierarchical-ALOHA+ALOHA strategy does not have these drawbacks. So, it performs better than the hierarchical-ALOHA+TDMA strategy.

The direct communication strategy using two channel tunable optical filter at the master has similar advantages as the hierarchical-ALOHA+ALOHA strategy in terms of reducing contention between the slaves for sub-master's channel. As slaves directly send their results to the master, duplication of result transfer (once from slaves to sub-masters and then from sub-masters to master) is avoided. Hence, using direct communication with tunable optic filters is a better strategy than the hierarchical-ALOHA+ALOHA strategy.

The following observations are noteworthy:

- direct-ALOHA strategy achieved its best performance when the degree of parallelism is close to 12;

- the hierarchical-ALOHA+ALOHA and direct communication with two channel tunable filter strategies achieved their best performance when the degree of parallelism is close to 24, twice that for direct-ALOHA;
- this is consistent with the analysis in Section 4.3 if $p \approx 1/12$ and $n \approx 12$ for result matrices of size 800×800.

The direct-TDMA strategy shows the best performance as the degree of parallelism increases. This is due to a total absence of channel contention between the slaves. Also, the overheads associated with the TDMA protocol decline with increasing parallelism. As already mentioned, for any simulation run the number of slots in a TDMA cycle is equal to the maximum number of slaves that can be spawned in any phase of task execution. With the parameters assumed, for a given simulation run, the difference between the maximum and minimum number of slaves spawned in different phases can be at most six. So, at most six time slots in a TDMA cycle can remain unutilized if all the slaves are trying to send their results simultaneously. As the degree of parallelism increases this number becomes a smaller fraction of the total number of time slots in a cycle. So, overall channel utilization increases with increasing parallelism. An added advantage of the direct-TDMA and the tunable filter strategies is that they do not require extra nodes to act as sub-masters.

The inflexibility of the TDMA protocol is illustrated by the fact that for low degrees of parallelism the overheads due to unutilized slots are significant and the direct-TDMA strategy has the worst performance of the five strategies. For high degrees of parallelism the overheads decline relatively. So, direct-TDMA shows the best performance. Also, all the simulations were carried out for results being 800×800 matrices. Matrices of this size require exactly four time slots to be transmitted. So, the problem of slot underutilization is non-existent.

The hierarchical-ALOHA+TDMA strategy appears to suffer from the drawbacks of both the TDMA and ALOHA protocols, namely increased cycle time and greater contention, respectively at higher degrees of parallelism. Hence, its performance is poorer than both hierarchical-ALOHA+ALOHA and direct-TDMA strategies. We also simulated a hierarchical communication protocol that used time division multiplexing at both the levels. Its performance was almost identical to that of the direct-TDMA strategy, and hence, has not been reported here. Such identical results further emphasize the effect of contention between the two sub-masters for the master's channel.

From these simulation results, it can be observed that the direct-ALOHA and the hierarchical-ALOHA+TDMA strategies demonstrate poor performance for varying degrees of parallelism and data size. Thus, these combinations are not suitable for building general-purpose heterogeneous parallel systems. So, in the remaining part of the paper, we eliminate these two strategies and restrict our study to the remaining three communication strategies.

6.4 Inflexibility of TDMA Protocol

In order to verify the fact that the performance of the direct-TDMA strategy is highly dependent on the result size, we compared this strategy with the hierarchical-ALOHA+ALOHA and the direct-with two channel tunable filter at master strategies for smaller N. These results are illustrated in Fig. 10 for $N = 500$. The hierarchical-ALOHA+ALOHA strategy has a better performance than the direct-TDMA strategy for degree of parallelism up to 32. For higher degrees of parallelism, the direct-TDMA strategy shows better performance than the hierarchical-ALOHA+ALOHA strategy. The direct communication strategy that uses two channel tunable filter at the master with ALOHA has the best performance for the entire range. The results of Fig. 10 can be explained as follows:

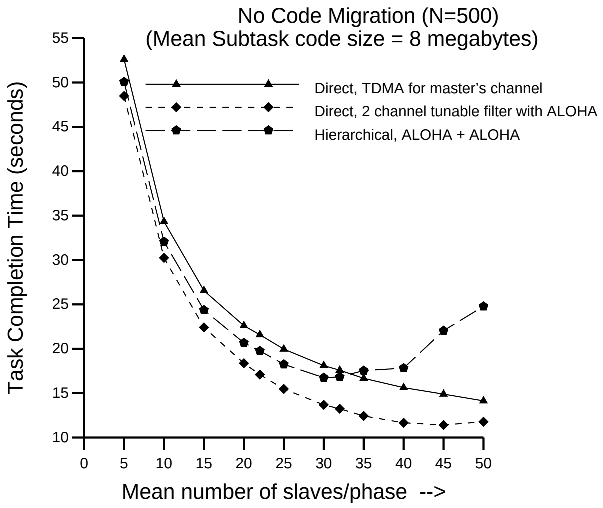

Figure 10: Performance of Hierarchical-ALOHA-ALOHA, Direct-TDMA and Direct-2 channel tunable filter with ALOHA strategies for smaller data/result sizes (N=500).

a slave needs approximately 1.56 TDMA time slots to send its result (a 500×500 matrix) to the master. So, every other time slot that a slave uses to send results is only 56% utilized, thus reducing the communication efficiency. The other two strategies do not have the problem of underutilized slots. Also, the hierarchical strategy leads to a reduction in the contention between the slaves when they have to send their results. Using tunable filters has the added advantage of the total absence of contention between the sub-masters for the master's channel that is present in hierarchical strategies.

Therefore, even though the direct-TDMA strategy may have the best performance in some situations, its performance is highly dependent on the nature of the application and the data size. The direct-TDMA strategy lacks flexibility as the slot duration and the number of slots in a TDMA cycle are fixed *a priori* at the time of protocol design. In contrast, the hierarchical-ALOHA+ALOHA and the direct-with two channel tunable filter strategies consistently perform well in all situations. Hence, the latter two strategies are suitable for general purpose heterogeneous parallel systems while the direct-TDMA strategy is not.

Fixed degree of parallelism (low) and varying subtask data size		
Data/Result size		
small	medium	large
all	1, 3, 4	3, 4

Fixed subtask data size and varying degree of parallelism		
Degree of parallelism		
low	medium	high
all	2, 3, 4, 5	2, 3, 4

Figure 11: Suitability of various communication topology–protocol combinations for different data sizes and degrees of parallelism. 1 = Direct-ALOHA for master's channel, 2 = Direct-TDMA for master's channel, 3 = Direct-2 channel tunable filter for master-ALOHA, 4 = Hierarchical-ALOHA+ALOHA, 5 = Hierarchical-ALOHA+TDMA.

It is interesting to note that for the hierarchical-ALOHA+ALOHA strategy the best performance is achieved when the degree of parallelism is equal to 32. This is greater than the corresponding value of 24 when the slave result matrices are of size 800×800. This implies that maximum throughput is achieved for $n \approx 16$ and $p \approx 1/16$ for 500×500 matrix results (compared to $p \approx 1/12$ for 800×800 matrix results) validating the assumption in Section 4.3 that p increases with increasing result size.

7 Summary of Evaluation and Design Choices

Based on the simulation results, the communication strategies suitable for different subtask data sizes and different degrees of parallelism in the applications are shown in Fig. 11. The direct-TDMA, direct-with 2 channel tunable filter for master-ALOHA and the hierarchical-ALOHA+ALOHA strategies are found to be suitable for applications with larger data/result size. However, as already shown in Fig. 10 and described in Section 6.3, the TDMA protocol lacks flexibility because the number of slots in a TDMA cycle and slot duration are fixed *a priori*. Hence, the direct-TDMA strategy is not suitable for building general-purpose parallel systems if the size of the result is variable.

The two-level hierarchical-ALOHA+ALOHA and the direct with 2 channel tunable filter for master strategies are independent of applications and result sizes. So, they have the flexibility that the direct-TDMA strategy lacks. The performance of the hierarchical-ALOHA+ALOHA strategy may degrade for high degrees of parallelism and large data sizes due to a dramatic rise in contention for the channels of the sub-masters and the master. This contention can be reduced by using multiple levels in the hierarchy with the number of children of interior nodes increasing as one goes further away from the master. However, unlike fat-trees [11], the communication bandwidth required by the channels at all the levels would be the same. This is because at higher levels of the hierarchy, the increase in the volume of results to be transferred (reflected by p) would be compensated by the reduced contention as fewer nodes (corresponding to n) will be contending for their parent's channel, supporting the assertion in Section 4.3 that for optimal performance, namely, $n \times p = constant$. This uniformity in the bandwidth of the home channels of the master, sub-masters, and slaves makes the hierarchical interconnection scalable and attractive.

The direct-with 2 channel tunable filter for master strategy has comparable performance to the hierarchical-ALOHA+ALOHA strategy. Tunable filters that can concurrently be tuned to multiple channels are available. The extra cost of such multi-channel tunable filters is not a significant drawback when compared to the overheads of the hierarchical strategy which needs sub-masters in addition to the master and slaves.

8 Conclusions

In this paper we have studied issues pertaining to computation and communication in heterogeneous parallel systems. Current high-speed networks cannot fully satisfy the high-bandwidth and low latency requirements of heterogeneous systems because they employ electronic network protocols. Developments in the area of optical networking, especially passive star-coupled optical interconnection, promise to meet the bandwidth and latency demands of heterogeneous systems. In this paper, we considered medium access strategy, contention resolution protocols and communication topology in an integrated manner. We developed five different communication topology–protocol combinations to work with passive star-coupled optical interconnection. All these topology–protocol combinations were evaluated for heterogeneous parallel systems with commonly used master-slave computation model.

It was observed that three topology–protocol combinations are capable of significantly reducing channel contention and delivering the best performance. These combinations are hierarchical interconnection with ALOHA protocol, direct communication with TDMA protocol and direct communication with tunable optical filters at the nodes. However, the TDMA protocol was found to be suitable for only a limited set of applications. The remaining two combinations were observed to deliver comparable performance for a wide range of applications. Therefore, the proposed hierarchical interconnection on a passive star-coupled optical network with ALOHA protocol is most suitable for heterogeneous parallel systems. The analysis and results presented in this paper provide guidelines for building large-scale heterogeneous systems to deliver high performance while minimizing communication overheads and channel contention. The simulation results lead us to believe that when the hierarchical interconnection is combined with the tunable optical filter approach, the levels in the hierarchy can be reduced, and performance can be enhanced. Such a combination of two efficient strategies may be interesting to explore.

References

[1] *IEEE 1596.4 IEEE Standard for High Bandwidth Memory Infrastructure based on SCI Signaling Technology (RamLink)*, December 1993.

[2] *Scalable Coherent Interface: ANSI/IEEE Standard 1596-1992*, August 1993.

[3] N. Abrahamson. Development of the ALOHANET. *IEEE Transactions on Information Theory*, IT-31:119–123, March 1985.

[4] E. A. Arnould, F. J. Bitz, E. C. Cooper, H. T. Kung, R. D. Sansom, and P. A. Steenkiste. The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers. In *Proceedings of the 3^{rd} International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS III)*, pages 205–216. ACM/IEEE, April 1989.

[5] 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 Communication*, 11(4):590–604, May 1993.

[6] M. Borella, B. Mukherjee, F. Jia, S. Ramamurthy, D. Banerjee, and J. Iness. Optical Interconnects for Multiprocessor Architectures Using Wavelength-Division Multiplexing. In *Proceedings of 27^{th} HICSS*, pages 499–508, January 1994.

[7] C. A. Brackett. Dense Wavelength Division Multiplexing Networks: Principles and Application. *IEEE Journal on Selected Areas in Communication*, 8(6):948–964, August 1990.

[8] K. W. Cheung, S. C. Liew, D. A. Smith, C. N. Lo, J. E. Baran, and J. J. Johnson. Simultaneous Five-Wavelength Filtering at 2.2 nm Wavelength Separation Using an Intergrated Optic Tunable Filter with Subcarrier Detection. In *Proceedings of ECOC, Gothenburg, Sweden*, volume 1, pages 312–315, 1989.

[9] K. W. Cheung, D. A. Smith, J. E. Baran, and B. L. Heffner. Multiple Channel Operation of an Integrated Acousto-optic Tunable Filter. *Electronics Letters*, 25:375–376, 1989.

[10] R. Chipalkatti, Z. Zhang, and A. S. Acampora. Protocols for Optical Star-Coupled Network Using WDM: Performance and Complexity Study. *IEEE Journal on Selected Areas in Communication*, 11(4):579–589, May 1993.

[11] Thinking Machines Corporation. *The Connection Machine CM5 Technical Summary.* Cambridge, MA, 1992.

[12] N. R. Dono, Jr. P. E. Green, K. Liu, R. Ramaswami, and F. F.-K. Tong. A Wavelength Divison Multiple Access Network for Computer Communication. *IEEE Journal on Selected Areas in Communication*, 8(6):983–994, August 1990.

[13] R. F. Freund, S. Natarajan, and V. K. Prasanna. Experiences in Using Heterogeneous Computing for Image Understanding. In *Proceedings of the Heterogeneous Computing Workshop*, pages 53–60. IEEE, April 1995. Held in conjunction with IPPS.

[14] R. F. Freund and H. J. Siegel. Heterogeneous Processing. *IEEE Computer*, pages 13–17, June 1993.

[15] D. Gustavson and Q. Li. Local Area MultiProcessor: the Scalable Coherent Interface. In S. F. Lundstrom, editor, *Defining the Global Information Infrastructure: Infrastructure, Systems, and Services*, volume 56, pages 131–160. SPIE Press, 1994.

[16] E. Haddad. Load Distribution Optimization in Heterogeneous Multiple Processor Systems. In *Proceedings of the 2^{nd} Workshop on Heterogeneous Processing*, pages 42–47. IEEE, 1993.

[17] D. A. Hensgen, L. Moore, T. Kidd, R. F. Freund, E Keith, M. Kussow, J. Lima, and M Campbell. Adding Rescheduling to and Integrating Condor with SmartNet. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–11. IEEE, April 1995. Held in conjunction with IPPS.

[18] The Institute of Electrical and Electronics Engineers. *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications – American National Standard ANSI/IEEE Standard 802.3*, 1990.

[19] Paragon XP/S Product Overview. Intel Corporation, 1991.

[20] M. A. Iqbal. Partitioning Problems in Heterogeneous Computer Systems. In *Proceedings of the 2^{nd} Workshop on Heterogeneous Processing*, pages 23–28. IEEE, 1993.

[21] A. Khokhar, V. K. Prasanna, M. Shaaban, and C.-L. Wang. Heterogeneous Supercomputing : Problems and Issues. In *Proceedings of the Workshop on Heterogeneous Processing*, pages 3–12. IEEE, March 1992.

[22] H.T. Kung, R. Sansom, S. Schlick, P. Steenkiste, M. Arnould, F. J. Bitz, F. Christianson, E. C. Cooper, O. Menzilcioglu, D. Ombres, and B. Zill. Network-Based Multicomputers : An Emerging Parallel Architecture. In *Supercomputing*, pages 664–673, 1991.

[23] D. J. Lilja. Experiments with a Task Partitioning Model for Heterogeneous Computing. In *Proceedings of the 2^{nd} Workshop on Heterogeneous Processing*, pages 29–35. IEEE, 1993.

[24] B. Mukherjee. WDM-Based Local Lightwave Networks Part I: Single-Hop Systems. *IEEE Network*, pages 12–27, May 1992.

[25] B. Mukherjee. WDM-Based Local Lightwave Networks Part II: Multihop Systems. *IEEE Network*, pages 20–32, July 1992.

[26] R. Prakash and D. K. Panda. Architectural Issues in Designing Heterogeneous Parallel Systems with Passive Star-Coupled Optical Interconnection. In *Proceedings of the IEEE International Symposium on Parallel Architecture, Algorithms and Networks*, pages 246–253, Kanazawa, Japan, December 1994.

[27] F. E. Ross. An Overview of FDDI: The Fiber Distributed Data Interface. *IEEE Journal on Selected Areas in Communication*, 7(7):1043–1051, September 1989.

[28] H. Schwetman. *CSIM Revision 16*. Microelectronics and Computer Technology Corporation, 3500 West Balcones Center Drive, Austin TX 78759, U.S.A., June 1992.

[29] G. C. Sih and E. A. Lee. A Compile-Time Scheduling Heuristic for Interconnection- Constrained Heterogeneous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, pages 175–187, February 1993.

[30] J.E. Smith, W.-C. Hsu, and C. Hsiung. Future General Purpose Supercomputer Architectures. In *Supercomputing*, pages 796–804. IEEE, November 1990.

[31] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1988.

[32] A. Varma, V. Sahai, and R. Bryant. Performance Evaluation of a High-Speed Switching System Based on the Fibre Channel Standard. In *Proceedings of the Second International Symposium on High Performance Distributed Computing*, pages 144–151, Spokane, Washington, July 1993. IEEE.

[33] R. J. Vetter, D. H.C. Du, and A. E. Klietz. Network Supercomputing : Experiments with a CRAY-2 to CM-2 HIPPI Connection. In *Proceedings of the Workshop on Heterogeneous Processing*, pages 87–92. IEEE, March 1992.

[34] M.-C. Wang, S.-D. Kim, M. A. Nichols, R. F. Freund, H. J. Siegel, and W. G. Nation. Augmenting the Optimal Selection Theory for Supercomputing. In *Proceedings of the Workshop on Heterogeneous Processing*, pages 13–21. IEEE, March 1992.

[35] Wolfram Research Inc. *Mathematica: A System for Doing Mathematics by Computer*, 1992.