Modeling Compiled Communication Costs in Multiplexed Optical Networks *

C. Salisbury and R. Melhem
Department of Computer Science
University of Pittsburgh
{salisbur,melhem}@cs.pitt.edu

Abstract

Improvements in optical technology will enable the construction of high bandwidth, low latency switching networks. These networks have many applications in massively parallel processing. However current circuit switching and packet switching techniques are not quite suitable for controlling such networks. Time division multiplexing (TDM) schemes can improve the performance of circuit switched optical interconnection networks by taking advantage of the locality of references present in the communication patterns. In this paper, we construct a model for the cost of compiled communications in circuit switched networks. We show how the cost is affected by the characteristics of the network and by the application's communication locality of references. We show how a compiler can use this information to choose the most appropriate multiplexing degree.

1. Introduction

Optical technologies can be used to build the high bandwidth, low latency interconnection networks required by high performance computing applications. Since it is impractical to build large networks by directly connecting all the attached nodes to each other, suitable network architectures and control techniques must also be developed to realize the full optical potential. The following two approaches have been used for designing networks.

- *Reducing direct connectivity. Packet switching* techniques send messages through the network to their final destinations via intermediate nodes. Routing information added to each message is processed by electronic circuitry, requiring optical signals to be converted to electronic signals (and possibly buffered) at each routing step. This leads to under-utilization of the full capability of an optical interconnection.

*This work is supported in part by NSF award MIP-9633729 and by AFOSR award F49620-93-1-0023DEF to the University of Pittsburgh

- *Sharing network resources. Circuit switching* techniques using time[5], space[13], or optical wavelengths[1] can provide direct connections between devices. However only a subset of all possible connections can be provided at any given time. This reduces the cost and size of the network, but increases the complexity of managing the network. This alternative is attractive for optical networks since all-optical paths from sources to destinations can fully exploit the high bandwidth of optics [4].

While time division multiplexing (TDM) is a general technique that can be used in any network, it is especially attractive in optical networks because of the large bandwidth available. An interconnection network built from optical components can provide communication bandwidth of 250 Gb/s or more[12], an amount that exceeds the requirements of any single processor. In TDM the optical bandwidth is shared via a sequence of sets of interconnections. The network automatically cycles through this sequence without intervention by the program, establishing each set of connections for a small time interval called a *time slot*. The length of a slot is chosen to be long enough to transmit a single message. For example, at 10 Gb/s a 30 ns time slot can be used to transmit four words of 64 bits. Each processor may communicate with another processor during the time slot when the network provides the connection it needs. The cycle can be short enough so that the path is again available when the processor has additional data to transmit. TDM requires the use of a global clock to synchronize the time slots for all the transmitters and receivers connected to the network. The actual data transmission, however, does not require a global clock as self-clocking can be used at the bit level[2].

The set of paths provided by a network is determined by the state of the various network components, such as switches, registers, frequency assignments, etc. The collective state of the components is referred to as the network state. The required network state can be determined as soon as the application's communication requests are known. Depending on the network design and application requirements, this network state may need to change dur-

ing program execution. Both determining and changing the network state are potential sources of communication delay. Techniques that determine the application's communication pattern and the required network states prior to program execution are called *compiled communication*, or *static* techniques [2, 10]. Techniques that use information gathered at program execution time to determine which paths the program needs are called *dynamic* techniques.

To develop a control strategy for a circuit switched optical network, we can apply the well known concepts of *locality of reference* and *working set*. Managing the set of paths in use is similar to managing pages of virtual memory[3]. Circuit switched interprocessor communication networks will have different locality characteristics than LAN communications[8] or memories[7, 11].

In section 2 of this paper we present a model of the communication pattern of a parallel application. In section 3, we model the cost to communicate over a circuit switched network. The model is developed for compiled communication where the application's requirements are known prior to program execution. We show when TDM can reduce communication costs in terms of network and application characteristics. To investigate the application factors that influence communication cost, in section 4 we quantify the concept of communication locality of reference and show how to identify the working set of communication paths used by an application. In section 5, we give particular attention to applications that consist of parallel loops. We show that communication cost and locality of reference are related in section 6.

2. The application model

To the interconnection network, an application appears as a sequence of requests for a network connection. These requests can originate on any node attached to the network. Each request can be characterized by its source, destination, message length, and arrival time. We model the requests using fixed length messages and represent arrival times by ordering messages in the sequence in which they are presented to the network. The network processes each request by providing a path connecting the message source and destination. We will represent the sequence of communication connection requests by $\mathcal{T} = P_1, P_2, \ldots$, where P_i represent the source/destination pair (s_i, d_i) for the i^{th} communication request.

We refine the model to express the looping structure common to parallel applications by letting $\mathcal{T}$ be the concatenation of the requests from a sequence of loops. Let P_{ij} be the path required by the j^{th} communication request in the i^{th} loop. Let L_i be the sequence of requests generated by one iteration of loop i, so that $L_i = P_{i1}, P_{i2}, P_{i3}, \ldots$. Let the number of iterations of loop i be r_i. We can describe the sequence of requests generated by this loop as $L_i^{r_i}$, indicating that the sequence L_i is concatenated to itself r_i times. The requests from an application with p loops can then be described by:

$$\mathcal{T} = L_1^{r_1}, L_2^{r_2}, \ldots, L_p^{r_p}$$

We will often be interested in the number of different paths used in loop i, and define this number to be l_i. We will also find it useful to deal with looping structures that have disjoint loops or distinct paths, as defined below.

Definition 1: We say that a looping structure has *disjoint* loops if each loop uses paths that are not used by any other loop. Specifically, if M_i contains all the paths in L_i and M_j contains all the paths in L_j, then $M_i \cap M_j = \emptyset$ for all $i \neq j$.

Definition 2: We say that a loop has *distinct* paths when each path in the loop is used exactly once in each iteration of the loop. Loops with distinct paths have $l_i = |L_i| = |M_i|$.

3. Cost of TDM compiled communication

When the communication requirements of an application are known at compile time, a compiler can determine the needs of different sections of the program and insert instructions for the hardware to establish the required circuits [10]. This is called *compiled communication*, and can be more efficient than having the hardware establish a circuit for each individual connection request while the program is running. The term *communication cost* refers to the delays a program will encounter when communicating in a circuit switched network.

A network that uses time division multiplexing will automatically cycle through a sequence of network states. Each state provides a set of paths over which messages can be sent. The number of states in the cycle is referred to as the *multiplexing degree* which we will represent by K. The set of K states that are contained in one cycle are referred to as the *multiplexed state*. We will assume that the network can provide any arbitrary set of m paths simultaneously. For example, networks such as the Benes, Clos, cross-bar, and WDM star can provide connections for any arbitrary permutation [6, 9]. Thus, if the network can provide a maximum of m paths in one state, the multiplexed state can contain up to Km paths. Note that $K = 1$ corresponds to a non-multiplexed network.

The compiler can use a greedy algorithm to create the required sequence of network states. The communication requests are first placed in sequential order. A network state is created for the first group of m different paths. Additional states are then created for successive groups of m different paths. We will call this algorithm the "m-path" method. If the number of paths in the application's working set is greater than m, this method will require frequent establishment of new network states.

Another method to satisfy the application's requirements is to use multiplexing to provide complete interconnection by dividing all possible paths into sets of size m, and then cycling through all these sets. This simulates a completely connected network, but may not perform well when the application requires only a small subset of these paths.

A third alternative is to combine the m path method with multiplexing by having the compiler establish K network states at once, and letting the network cycle through these states without further direction from the program. Network performance will be affected by our choice of the value of K. If K is too small, communication requests will be delayed often while a network state with the required connection is being established. If K is too large, communication requests may be delayed while the network provides connections that are not needed during a particular phase of program execution. The optimal choice of K will reflect a balance between the network's performance characteristics and the application's path requirements. We will assume that the network can alter the value of K during program execution.

We next develop a model of communication cost for an application with 1) an arbitrary sequence of requests, 2) a single loop, and finally 3) a sequence of loops.

3.1. The communication cost model

Establishing a network state during program execution requires only processor synchronization (to avoid inconsistent states) and loading the network state. We consider the time to load the state of the network components to be constant regardless of the multiplexing degree, and that, over time, all processors will encounter the same amount of delay at synchronization points. Thus the delay caused by the establishment of a new network state is modeled as a fixed value, E_f.

When time division multiplexing is used, a program will be delayed when it attempts to use a path that is in the multiplexed state but is not established at the time of the request. On average, the length of the delay will be proportional to the degree of multiplexing. In the worst case, we could apply this delay to every communication request. Each request would have to wait for the network to move through the cycle before the required path became available. However, in a parallel system it is reasonable to assume that m requests will be generated and processed together, and thus the average total access cost required to handle q requests with multiplexing degree K is $A_m \frac{q}{m} \frac{K}{2}$ where A_m is the length of a time slot.

To compute the cost of a sequence of communication requests, $\mathcal{T}$, we first apply the m-path method to create a sequence of network states. Let s be the number of states created. In a non-multiplexed network, the total cost of communications is simply sE_f. We would like to compare this to the cost in a multiplexed network.

To compute the cost in a multiplexed network, we will combine a group of u consecutive network states into a single multiplexed state. The sub-sequence of $\mathcal{T}$ representing the consecutive requests satisfied by this multiplexed state is called a *partition* of $\mathcal{T}$. There are $n = s/u$ such partitions. The i^{th} partition is referred to as T_i and we have $\mathcal{T} = T_1, T_2, \ldots, T_n$. When the u states of partition T_i are multiplexed together with degree K, the cost for the partition is

$$E_f + A_m \frac{|T_i|}{m} \frac{K}{2} \quad . \tag{1}$$

Note that the degree of multiplexing (K) may be different from the number of states that are multiplexed together (u). This may occur when the multiplexed states share common paths, since a given path has to appear in a multiplexed state only once.

In general, if $N_1, \ldots, N_u$ are network states that are to be combined into one multiplexed state, N', then the required multiplexing degree K is $\left\lceil \frac{|\cup_{j=1}^{u} N_j|}{m} \right\rceil$. We can find the total cost of providing $\mathcal{T}$ in a multiplexed network by determining the value of K for each partition and summing the costs from expression (1) for all s/u partitions. The total cost of $\mathcal{T}$ in the multiplexed network is

$$\frac{s}{u} E_f + A_m \sum_{i=1}^{s/u} \left(\frac{|T_i|}{2m} \left\lceil \frac{|\cup_{j=u(i-1)+1}^{ui} N_j|}{m} \right\rceil \right) \quad . \tag{2}$$

We can compare this expression to the cost in the non-multiplexed network to determine when multiplexing reduces the total cost of communication. We rearrange the inequality to place all the network parameters on one side, and call the resulting term α. We find that multiplexing all of $\mathcal{T}$ in groups of u states reduces cost when:

$$\alpha = \frac{E_f}{A_m/2m} > \frac{\sum_{i=1}^{s/u} \left(|T_i| \left\lceil \frac{|\cup_j N_j|}{m} \right\rceil \right)}{s - \frac{s}{u}}. \tag{3}$$

The right side of inequality (3) is determined by the way the sequence of communication requests can be grouped into network states of size m. Since the left hand side is constant for a given network, α provides a threshold by which we can judge the application to determine when multiplexing $\mathcal{T}$ by combining groups of u states reduces cost.

In the special case where the degree of multiplexing is large enough to multiplex together all the states required by $\mathcal{T}$, there will be just one partition. If t is the number of different paths used in $\mathcal{T}$, this occurs when $K = \lceil \frac{t}{m} \rceil$. We will refer to this as *one-partition multiplexing*, and to this value of K as the *one-partition multiplexing degree*. We know that in this case the establishment costs will be

minimized and inequality (3) can be reduced to:

$$\alpha > \frac{|\mathcal{T}|\lceil \frac{t}{m} \rceil}{s-1} \tag{4}$$

To make further statements about the benefit of multiplexing we must know more about the structure of $\mathcal{T}$. We begin the analysis in the following section by considering the communication pattern that arises from a single loop.

3.2. Communication cost in a single loop

Assume a single loop, L^r, with $l \leq |L|$ different paths and a network where $m < l$. Because the communication pattern repeats itself, we can restate inequality (4) in terms of the characteristics of a single loop iteration. We find that for one-partition multiplexing to reduce costs below the non-multiplexed case, it is sufficient that:

$$\alpha > \frac{\frac{m}{l}\lceil \frac{l}{m} \rceil |L|}{1 - \frac{m}{rl}} \tag{5}$$

Since $r \geq 2$ and $l > m$, one-partition multiplexing will reduce costs whenever $\alpha > 4|L|$.

We can determine the optimal degree of multiplexing for a loop that has distinct paths. In this case, the multiplexing degree is equal to the number of network states, u, that are combined. We can substitute values for α, s, and K into expression (2) to find the total cost:

$$\text{Cost}(L^r) = \begin{cases} \frac{A_m|\mathcal{T}|}{2m}\frac{\alpha}{m} & \text{if } K = 1 \\ \frac{A_m|\mathcal{T}|}{2m}(\frac{\alpha/m}{K} + K) & \text{if } 1 < K < \lceil \frac{l}{m} \rceil. \end{cases} \tag{6}$$

When $K = \lceil \frac{l}{m} \rceil$ we require only one network state. Assuming that l is a multiple of m and letting $u = s$, $|\mathcal{T}| = rl$, and $K = \frac{l}{m}$, we find:

$$\text{Cost}(L^r) = \frac{A_m|\mathcal{T}|}{2m}(\frac{\alpha/m}{rK} + K) \quad \text{if } K = \lceil \frac{l}{m} \rceil \tag{7}$$

Most of the reduction in establishment costs occurs with the first few degrees of multiplexing. Thus, the optimal degree of multiplexing may be less than the one-partition multiplexing degree. We can see how network characteristics and multiplexing degree affect cost with the example in Figure 1. We consider the cost of a single loop with a fixed number of distinct paths, l, running on a network with fixed values of α and m. In the figure, we assume r is very large, $\lceil \frac{l}{m} \rceil = 8$, and consider three networks with values of α/m equal to 8, 20, and 32. When $\alpha/m = 8$, the minimum cost occurs at $K = 3$ independent of the value of r. When $\alpha/m = 20$, the optimal choice depends on the number of iterations as we have just seen. When $\alpha/m = 32$, one-partition multiplexing reduces cost even when $r = 2$.

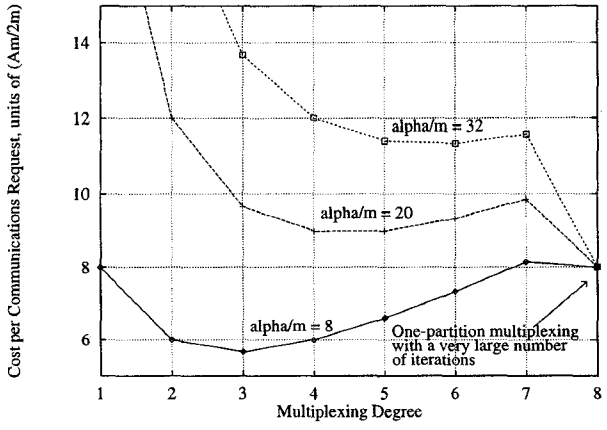

Figure 1. Cost of a multiplexed loop.

Thus, to determine the optimal multiplexing degree we must consider the relationship between network and application characteristics. The following Theorem whose proof is provided in [14] formalizes the conclusions.

Theorem 1: For a loop with l distinct paths and a network with $\frac{\alpha}{m} > 4$, one-partition multiplexing is always optimal (has the lowest communication cost) when $\lceil \frac{l}{m} \rceil \leq \sqrt{\frac{\alpha}{m}}(1 + \frac{1}{\sqrt{2}})$ and is never optimal when $\lceil \frac{l}{m} \rceil \geq 2\sqrt{\frac{\alpha}{m}}$. If multiplexing the entire loop isn't optimal, then the optimal multiplexing degree is $\sqrt{\alpha/m}$. □

3.3. Combining loops with multiplexing

When an application has several loops, we can continue to increase the degree of multiplexing beyond that required by a single loop. When a single network state provides the total number of paths required by two adjacent loops, we can again reduce state establishment costs. The increased multiplexing degree will increase path access costs for both loops, however. When all the loops have the same number of paths and the loops are disjoint, we can show [14] that multiplexing will reduce cost when $\alpha > |\mathcal{T}|\lceil \frac{l}{m} \rceil$. Unlike inequality (5), this threshold for combining loops is affected by the number of iterations of each loop. Hence it can be much larger than the threshold developed for a single loop.

3.4. Application characteristics and cost

In the same sense that α provides a characterization of the network independent of the application, we would like to characterize the application independently from the network. To understand how applications might be characterized, consider the cost of the following two loops with distinct paths. Loop 1 has $m + 1$ paths and r iterations. Let q be any integer and let loop 2 have $q(m + 1)$ paths and r/q

iterations. Both loops require $s = \frac{r(m+1)}{m}$ network states and, without multiplexing, have the same cost. However, the one-partition multiplexing degree and cost of loop 1 are less than the one-partition multiplexing degree and cost of loop 2. We would like to define a metric that encapsulates this characteristic of an application.

When the application consists of a single loop, the number of different paths used may be a reasonable measure of communication locality. However, this may not be an appropriate measure for applications that consist of a series of loops, or for a non-repeating sequence of communication requests. In the next section we develop a more general metric that incorporates the notions of temporal and spatial locality of reference and show how it is related to communication cost.

4. Modeling application locality

A communication pattern that is highly local is desirable for two reasons. First, high locality suggests that only a "small" number of different paths are required so that the application doesn't use many network resources at any one time. This loosely corresponds to the notion of high spatial locality. Thus, the basis of our locality metric is a logical grouping of communication requests into *partitions* that can be created in any arbitrary manner. Second, good locality also suggests that path utilization should be high, which means the paths provided should be reused often. This loosely corresponds to the notion of temporal locality. We combine these concepts of spatial and temporal locality in our locality metric. Small partition sizes coupled with high reuse of the paths in a partition means high locality. The choice of metric will necessarily reflect a balance between these two. That is, a reduction of spatial locality due to an increase in partition size can be offset by a gain in temporal locality due to an increase in the amount of reuse.

Recall that the sequence $\mathcal{T}$ can be broken into n sub-sequences such that $\mathcal{T} = T_1, T_2, \ldots, T_n$. Each subsequence T_i is a partition of $\mathcal{T}$. The set of these partitions forms a *partitioning* of $\mathcal{T}$, denoted $\mathcal{P}(\mathcal{T})$. Let M_i be the set of all the different paths used in the partition T_i. Since a path may be used more than once in T_i, then $|T_i| \geq |M_i|$, where $|T_i|$ and $|M_i|$ are the number of paths in T_i and M_i, respectively. We define spatial locality of a partitioning to be inversely proportional to the average number of different paths used in its partitions. Fewer paths per partition means greater spatial locality. That is,

$$\text{Spatial Locality } (\mathcal{P}(\mathcal{T})) = \frac{1}{(1/n)\sum_{i=1}^{n} |M_i|} \quad (8)$$

Temporal locality is expressed in terms of the reuse of paths within a partition. For partition T_i, the path reuse is $\frac{|T_i|}{|M_i|}$. To encourage the formation of partitions with high path reuse, we use the root-sum-of-squares of this reuse to weight the temporal locality measure strongly toward such partitions. Thus, the temporal locality of a partitioning is defined as:

$$\text{Temporal Locality } (\mathcal{P}(\mathcal{T})) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{|T_i|}{|M_i|}\right)^2} \quad (9)$$

The communication locality of reference is defined as the product of these two measures.

$$\text{Locality}(\mathcal{P}(\mathcal{T})) = \frac{\sqrt{\sum(\frac{|T_i|}{|M_i|})^2}}{\sum |M_i|} \quad (10)$$

There are many ways of forming partitions. For example, all communications can be placed in a single partition, the m-path algorithm can be used, or the partitions can be completely arbitrary. In our notation we will represent a sequence of communications requests as a sequence of numbers where each number represents a single use of a particular path. For clarity, the examples given in this paper will use small sequences and small values of m. It should be clear, however, that the concepts presented apply as well to arbitrarily large sequences and values of m.

Example 1: The sequence $\mathcal{T}$ = 1,2,2,1 represents the use of path 1, followed by path 2, followed by the reuse of paths 2 and 1. A partitioning of this sequence could be $T_1 = 1, 2, 2$ and $T_2 = 1$. In this case, $M_1 = \{1, 2\}$ and $M_2 = \{1\}$. Using m-path partitioning with $m = 1$, we would have $T_1 = 1$, $T_2 = 2, 2$, $T_3 = 1$, and $M_1 = \{1\}$, $M_2 = \{2\}$, $M_3 = \{1\}$. □

We define the locality of a sequence to be the greatest locality attainable from any partitioning of the sequence. Each of the partitions from the optimal partitioning contains a working set of communication paths. In the following lemma, we show that we can always increase the locality of a partitioning if we can divide one of the partitions into two pieces which have no paths in common. Lemma 2 then describes when locality is increased by joining adjacent partitions. The proofs of these lemmas are straightforward from the definition of locality.

Lemma 1: Let $\mathcal{P}(\mathcal{T})$ be a partitioning of $\mathcal{T}$ with partitions $T_1, \ldots, T_i, \ldots, T_n$ and let $\mathcal{P}'(\mathcal{T})$ be a partitioning formed from $\mathcal{T}$ by splitting partition T_i into T_{i1}, T_{i2} so that $\mathcal{P}'(\mathcal{T}) = T_1, \ldots, T_{i1}, T_{i2}, \ldots, T_n$. Let M_{i1} and M_{i2} be the set of paths corresponding to T_{i1} and T_{i2}, respectively. If $M_{i1} \cap M_{i2} = \emptyset$ then Locality$(\mathcal{P}'(\mathcal{T}))$ > Locality$(\mathcal{P}(\mathcal{T}))$.

Lemma 2: Let $\mathcal{P}(\mathcal{T})$ be a partitioning which has adjacent partitions T_i and T_{i+1}. Let M_i and M_{i+1} be the corresponding sets of paths. Let $\mathcal{P}'(\mathcal{T})$ be a partitioning identical to $\mathcal{P}(\mathcal{T})$ except that T_i and T_{i+1} are joined into a single partition. If $M_i \subseteq M_{i+1}$, then Locality$(\mathcal{P}'(\mathcal{T}))$ > Locality$(\mathcal{P}(\mathcal{T}))$ whenever

$$\frac{2|T_{i+1}|}{|T_i|} \geq \left(\frac{|M_{i+1}|}{|M_i|}\right)^2 - 1 \qquad \square$$

When $M_i = M_{i+1}$ the condition of Lemma 2 is always met and we will always increase locality by combining adjacent partitions that use the same paths. When adjacent partitions use almost the same set of paths ($M_i \subset M_{i+1}$ and $|M_i| \approx |M_{i+1}|$), we should consider merging the partitions even when the sequence lengths are very different ($|T_{i+1}| \ll |T_i|$). When the numbers of paths used by these partitions are very different ($|M_{i+1}| > |M_i|$) we should consider merging the partitions only when the partition with more paths has many more requests ($|T_{i+1}| \gg |T_i|$).

In the next section we again focus on looping applications, beginning with loops that are disjoint. We will determine when the greatest locality is obtained by placing all requests from each loop into a single partition. This partitioning corresponds to our intuitive notion that each loop forms a working set of paths. We will then consider loops that are not disjoint, first looking at two loops and then extending the discussion to any number of loops.

5. Locality of looping sequences

We define the partitioning that places all requests from each loop into a separate partition to be the *natural partitioning* of the application and refer to it as $\mathcal{P}_{\mathcal{N}}$. Intuitively, we expect this to be the partitioning with the greatest locality as long as the number of iterations of each loop is sufficiently large and the loops are sufficiently disjoint. To quantify these two conditions for general loop structures is rather complicated, if at all possible. Thus, in the next sections we will consider only special loop structures.

5.1. Sequences from disjoint loops

We can place an upper bound on the number of loop iterations needed to ensure that the natural partitioning of a sequence of disjoint loops has the maximum locality. The bound is presented in the following theorem whose proof is provided in [14].

Theorem 2: Let $\mathcal{T}$ be a sequence of communication requests from an application $\mathcal{T} = L_1^{r_1}, \ldots, L_p^{r_p}$ with p disjoint loops, and let k_i be the maximum number of times any path is used in a single iteration of loop i. Then the natural partitioning of the sequence has locality greater than any other partitioning when:

$$r_i \geq \frac{(k_i l_i)^2}{|L_i|} \text{ for all } i \qquad (11)$$

For loops with distinct paths, $k_i = 1$. This requires $r_i \geq l_i$ for $i = 1, \ldots, p$. $\square$

Sequence	Method	Sets of Paths	Locality
$(1,2,3,3)^{20}$	natural	{1,2,3},{4,5,6}	6.28
$(4,5,6,6)^{20}$	one path	{1},{2},{3},..., {4},{5},{6},...	0.13
$(1,2,3,4)^{30}$	natural	{1,2,3,4},{5,6,7,8}	3.95
$(5,6,7,8)^{10}$	one path	{1},{2},{3},{4},..., {5},{6},{7},{8},...	0.08
$(1,2,3,4)^{20}$	natural	{1,2,3,4},{5,6,7,8}	3.54
$(5,6,7,8)^{20}$	one path	{1},{2},{3},{4},..., {5},{6},{7},{8},...	0.08

Table 1. Partitioning disjoint loops.

Table 1 shows several looping sequences, the sets of paths required by the natural and one-path partitionings, and the resulting locality. For these sequences, the number of iterations is large enough to ensure that the natural partitioning has the greatest locality. The locality of the one path partitioning is very low because there is little or no path reuse, so that temporal locality is low. The large number of partitions means that the sum of the partition sizes will be large, making spatial locality low as well.

5.2. Locality of two non-disjoint loops

Lemma 2 can be applied when the paths used by one loop are a subset of paths used by an adjacent loop. In this section we show how overlap between adjacent loops can be handled.

The locality when the paths from two adjacent loops are combined into a single partition is determined by the sequence length $|\mathcal{T}|$, the number of distinct paths in each loop (l_1 and l_2), and the amount of overlap between these sets of paths. We define the overlap v to be the number of times paths reappear in different partitions. The overlap between L_1 and L_2 is thus $v = l_1 + l_2 - |M_1 \cup M_2|$. Note that when the total number of requests is fixed the locality of the natural partitioning still depends on r_1 and r_2, while the locality of the combined partitioning depends only on the total sequence length. We define $r_{1,min}$ to be the value of r_1 which gives the minimum locality value possible for the natural partitioning.

We can most easily investigate the relationship between the locality of a combined partitioning and the natural partitioning by considering two loops with the same number of distinct paths, so that $l = l_1 = l_2 = |L_1| = |L_2|$. We assume the loops have l' paths in common, so $v = l'$. Since $|L_1| = |L_2|$, the length of the sequence is constant when the total number of iterations is constant. We can compute the ratio of the locality of a combined partitioning to the locality of the natural partitioning to be:

$$\text{Locality Ratio} = \frac{2}{(2-\frac{v}{l})^2\sqrt{1+\frac{2r_1}{r_1+r_2}(\frac{r_1}{r_1+r_2}-1)}} \qquad (12)$$

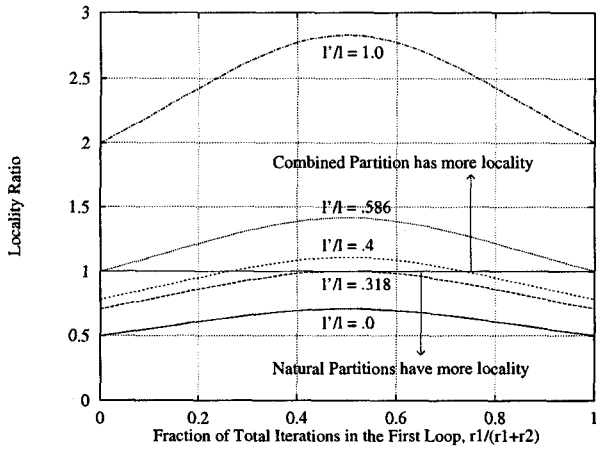

Figure 2. Combining two partitions.

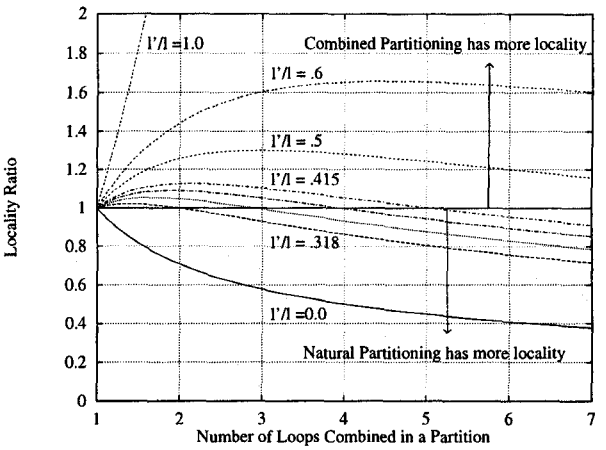

Figure 3. Combining many partitions

This ratio is plotted in Figure 2. The graph is symmetric because of the assumption that $|L_1| = |L_2|$. When $|L_1| \neq |L_2|$ the graph becomes skewed, with the peak moving toward the left when $|L_1| < |L_2|$. The amount of overlap required for the combined partitioning to have the greatest locality is stated in the following Lemma. The proof is given in [14].

Lemma 3: Consider an application consisting of two loops which use an equal number of different paths, so that $l_1 = l_2$. Assume the loops have l' paths in common. The natural partitioning of this application will always have locality greater than the combined partitioning when $(l'/l) \leq 0.318$. The combined partitioning of this application will always have locality greater than the natural partitioning when $(l'/l) \geq 0.586$.

5.3. Locality of multiple non-disjoint loops

We can investigate the effect of combining more than two non-disjoint loops if we assume that all loops have the same number of distinct paths and the same number of iterations. Consider an application with p such loops with r iterations and l paths. Let M_i be the set of paths used by loop i. The locality of the natural partitioning of the p loops is $\frac{r}{l\sqrt{p}}$. Create a new partitioning by combining u adjacent loops together into a single partition. We can generalize the definition of overlap v used in the last section and note that the number of times paths are reused among the u loops is $v = \sum_{i=1}^{u} l_i - |\cup_{i=1}^{u} M_i|$. If we further assume that the overlap is the same for each group of u loops, we find the locality of the new partitioning to be:

$$\frac{r}{l\sqrt{p}}\left(\frac{u\sqrt{u}}{(u-\frac{v}{l})^2}\right) \qquad (13)$$

Sequence	Method	Sets of paths	Locality
$(1,2,3,4)^{20}$	natural	{1,2,3,4}{3,4,5,6}	3.54
$(3,4,5,6)^{20}$	combined	{1,2,3,4,5,6}	4.44
$(1,2,3,4)^{20}$	natural	{1,2,3,4}{4,5,6,7}	3.54
$(4,5,6,7)^{20}$	combined	{1,2,3,4,5,6,7}	3.27

Table 2. Partitioning overlapping loops.

One application with the same overlap in each group of u loops has l' paths common to all loops, and $l - l'$ paths in each loop which are disjoint. In this case, we find $v = (u-1)l'$. For this application, the locality of the combined partitioning relative to the natural partitioning is shown in Figure 3. When there is little or no overlap, the natural partitioning always has the greatest locality. As the amount of overlap increases, locality is increased by combining loops into a single partition. For a given amount of overlap, we can determine the number of loops to be combined to attain the optimal locality. For example, if the application is $(1,2,3,4)^r(1,2,5,6)^r(1,2,7,8)^r$ then $l'/l = .5$. From Figure 3 we can see that we get the greatest locality when all three loops are combined into a single partition. Table 2 shows examples of partitionings for loops that overlap.

In this section we have argued that for sufficiently disjoint loops and a sufficiently large number of repetitions, the natural partitioning of an application provides the greatest locality of any partitioning, so that the communication working set is defined by loop boundaries. From the cost model, we found that communication costs can often be minimized by multiplexing to the degree that provides all the paths required by a loop. This similarity suggests that locality is an application characteristic that can be useful in determining the working set of paths that a network must provide in order to minimize communication costs. In the next section, we develop similar properties that partially

order applications by cost and by locality, and look at some example sequences.

6. Locality and the cost of communication

From the definitions of cost and locality we can develop properties for partially ordering applications, assuming the natural partitioning and one-partition multiplexing is used in each loop. The properties show that similar sequences of equal length will often be ordered in a corresponding way by cost and locality. Equality or inequality in the cost properties is determined by the network's value of m.

Property 1: Given two disjoint sequences L_1 and L_2 and a fixed number of iterations r:

- $\text{Cost}(L_1^r, L_2^r) \leq \text{Cost}((L_1, L_2)^r)$
- $\text{Locality}(L_1^r, L_2^r) > \text{Locality}((L_1, L_2)^r)$

Locality always increases, and cost may decrease, with disjoint sets of paths in separate partitions. Cost may decrease even when there is some overlap between the sets of paths. For example, the natural partitioning of $(1,2,3,4)^{20}(5,6,7,8)^{20}$ has more locality than the natural partitioning of $(1,2,3,4,5,6,7,8)^{20}$, and may cost less.

Property 2: Given two loops, $L_1^{r_1}$ and $L_2^{r_2}$ where $r_1|L_1| = r_2|L_2|$ and $l_1 < l_2$:

- $\text{Cost}(L_1^{r_1}) \leq \text{Cost}(L_2^{r_2})$
- $\text{Locality}(L_1^{r_1}) > \text{Locality}(L_2^{r_2})$

For an equal number of requests, a loop with fewer paths will have greater locality and may cost less than a loop with more paths.

Property 3: Given two sequences, L_1 and L_2, and iteration values r_1, r_2, r_1', and r_2' where $r_1 < r_1'$, $l_1 < l_2$, and $r_1|L_1| + r_2|L_2| = r_1'|L_1| + r_2'|L_2|$:

- $\text{Cost}(L_1^{r_1}, L_2^{r_2}) \leq \text{Cost}(L_1^{r_1'}, L_2^{r_2'})$
- $\text{Locality}(L_1^{r_1}, L_2^{r_2}) > \text{Locality}(L_1^{r_1'}, L_2^{r_2'})$ if $r_1' > r_{1,min}$.

When loops have a different number of paths, the distribution of requests between two loops affects cost and locality. As the proportion of requests in the loop with fewer paths increases, cost may decrease. Above a threshold value, locality increases. The need for the threshold is shown by an example. Note that $(1,2)^{14}(3,4,5,6)^{28}$ has less locality than $(1,2)^{10}(3,4,5,6)^{30}$, and it also costs less.

In general, looping sequences with disjoint sets of paths and a sufficient number of iterations will be placed in the same partial order by both locality and cost.

Request Sequence	Locality	Cost	
		Non-mpx	Mpx
$(1,2)^{80}$	40.0	1600	330
$(1,1,1,2)^{40}$	40.0	800	330
$(1,2,3,3)^{40}$	17.8	1200	500
$(1,2)^{40}(3,4)^{40}$	14.1	1600	340
$(1,2,3,4)^{40}$	10.0	1600	650
$(1)^{70}(2,3,4,\ldots,9,10)^{10}$	7.07	910	840
$(1,2,3,4,5)^{32}$	6.40	1600	830
$(1,2)^{50}(3,4,5,6,7,8)^{10}$	6.37	1600	580
$(1,2,3,3)^{20}(4,5,6,6)^{20}$	6.28	1600	500
$(1,2,3,4,5,6)^{27}$	4.44	1600	1290
$(1,2,3,4)^{20}(5,6,7,8)^{20}$	3.54	1600	660
$(1,2,3,4,5,6,7,8)^{20}$	2.50	1600	1290
$(1,2,3,4,\ldots,9)^{18}$	1.96	1600	1450
$(1,2,3,4,\ldots,10)^{16}$	1.60	1600	1610

Table 3. Locality and cost of several request sequences.

When network states are of fixed size, we can broaden the interpretation of the path notation used in the previous examples. Rather than representing a single path, an integer can represent an entire network state of m paths. All such states must be disjoint. A loop requiring m paths in each of two network states can therefore be represented as $(1,2)^r$. This loop can be contained in a single multiplexed network state with a multiplexing degree of two. For example, consider m processors arranged as a $\log m$-dimensional binary hypercube. The integer i can represent the set of m paths used for communication over dimension i. These sets of paths are disjoint. The notation $(1,2,\ldots,\log m)^r$ then represents the pattern of m processors communicating over each dimension in succession in a loop executed r times. Similarly, $(1,2,3,4)^r$ can represent a pattern where each of m processors in a two dimensional torus communicates once to its nearest neighbors in each dimension.

Table 3 shows the locality, non-multiplexed cost, and multiplexed cost for several example communication patterns. The ordering obtained from the locality measure in expression (10) follows the intuitive notion of locality. For computing costs, α was set at 10. Since very little of the multiplexing cost is state establishment cost, changing the value of α has very little effect on the total multiplexed costs. Decreasing α reduces the non-multiplexed costs, so that multiplexing reduces cost only for the most local sequences. Increasing α increases the non-multiplexed costs and the advantage of multiplexing.

Although the non-multiplexed cost can be low for some sequences with good locality, other sequences with good locality incur a large non-multiplexed cost. Additionally, sequences with equivalent non-multiplexed costs can have very different locality characteristics. There is a much stronger relationship between locality and cost when mul-

tiplexing is used. Multiplexing provides lower cost for sequences with greater locality. This ability to exploit locality is true for any value of α. This means that high locality is associated with low multiplexed cost regardless of the network characteristics.

7. Conclusions

The cost of controlling a interconnection network is a major concern in circuit switched optical networks since this cost can be very high relative to the cost of data transmission. Time division multiplexing is a technique that can be used to reduce the cost of establishing the required circuits. We analyzed the cost of circuit switched communications when the communication pattern can be determined by the compiler. A similar technique can be applied to dynamically controlled networks. We considered applications that contain parallel loops and quantified the cost reduction possible with multiplexing in terms of network and application characteristics. This information can be used by a compiler to determine the optimal degree of multiplexing. Costs can often be reduced by multiplexing together all the network states that provide the paths used in a loop.

Locality is a characteristic of the application that, in some cases, can be understood before the application is executed. While the locality measure provides an objective basis for determining the working set of communication paths, the real significance is the correspondence between locality and cost. Locality can be used as a tool to predict communication performance of an application without regard to the network on which the application will be implemented. The programmer can take expected communication performance into consideration when choosing the most appropriate algorithm to solve a specific problem.

The exact relationship between locality and communication cost depends on many factors such as the network parameters, the order of the communication requests and the number of paths used. The communication cost in a multiplexed network generally corresponds better to the locality of the application than does the cost in a non-multiplexed network. Thus multiplexing can be a valuable tool to exploit the locality of references and reduce communication costs in optical networks.

References

[1] C. A. Brackett. Dense wavelength division multiplexing networks: Principles and applications. *IEEE Journal on Selected Areas of Communications*, 8(6):948–964, August 1990.

[2] F. Cappello and C. Germain. Toward high communication performance through compiled communications on a circuit switched interconnection network. *Proceedings of the First IEEE Symposium on High-Performance Computer Architecture*, pages 44–53, January 1995.

[3] D. M. Chiarulli, S. P. Levitan, R. G. Melhem, and C. Qiao. Locality based control algorithms for reconfigurable optical interconnection networks. *Applied Optics*, 33:1528–1537, March 1994.

[4] D. M. Chiarulli, S. P. Levitan, R. G. Melhem, J. P. Teza, and G. Gravenstreter. Multiprocessor interconnection networks using partitioned optical passive star (POPS) topologies and distributed control. In *Proceedings of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, pages 70–80. IEEE, April 1994.

[5] I. Chlamtac and A. Ganz. Channel allocation protocols in frequency-time controlled high speed networks. *IEEE Trans. on Communications*, 36(4):430–440, April 1988.

[6] P. Dowd. Random access protocols for high-speed interprocessor communications based on an optical passive star topology. *IEEE Journal of Lightwave Technology*, 9(6):799–808, 1991.

[7] K. Grimsrud, J. Archibald, R. Frost, and B. Nelson. On the accuracy of memory reference models. In *Computer Performance Evaluation, Modeling Techniques and Tools. 7th International Conference Proceedings*, pages 369–388. Springer-Verlag, May 1994.

[8] N. Gulati, C. Williamson, and R. Bunt. LAN traffic locality: Characterization and application. In *Local Area Network Interconnection. Proceedings of the First International Conference*, pages 233–250. Plenum, October 1993.

[9] K. Hwang. *Advanced Computer Architecture*. McGraw-Hill, New York, NY, 1993.

[10] J. Li and M. Chen. Compiling communication-efficient programs for massively parallel machines. *IEEE Transactions on Parallel and Distributed Systems*, 2(3):361–375, 1991.

[11] D. T. Michel and W. C. Hobart, Jr. Toward a unified model of program behavior. *Performance Evaluation*, 20(20):27–44, May 1994.

[12] P. Prucnal, I. Glesk, and J. Sokoloff. Demonstration of all-optical self-clocked demultiplexing of tdm data at 250gb/s. In *Proceedings of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, pages 106–117. IEEE, April 1994.

[13] C. Qiao and R. Melhem. Reconfiguration with time division multiplexing MINs for multiprocessor communications. *IEEE Transactions on Parallel and Distributed Systems*, 5(4):337–352, 1994.

[14] C. Salisbury and R. Melhem. Modeling communication costs in multiplexed optical switching networks. Technical Report 96-22, Department of Computer Science, University of Pittsburgh, 1996.

[15] X. Yuan, R. Melhem, and R. Gupta. Compiled communication for all-optical TDM networks. In *Supercomputing '96*. IEEE, November 1996.