Partitioned Optical Passive Star (POPS) Multiprocessor Interconnection Networks with Distributed Control

Donald M. Chiarulli, Steven P. Levitan, *Senior Member, IEEE,* Rami P. Melhem, James P. Teza, and Greg Gravenstreter

***Abstract*—This paper presents a partitioned optical passive star (POPS) interconnection topology and a control methodology that, together, provide the high throughput and low latency required for tightly coupled multiprocessor interconnection applications. The POPS topology has constant and symmetric optical coupler fanout and only one coupler between any two nodes of the network. Distributed control is based on the state sequence routing paradigm which multiplexes the network between a small set of control states and defines control operations to be transformations of those states. These networks have highly scalable characteristics for optical power budget, resource count, and message latency. Optical power is uniformly distributed and the size of the system is not directly limited by the power budget. Resource complexity grows as $O(n)$ for the couplers, $O(n\sqrt{n})$ for transceivers, and $O[\sqrt{n}\log(n)]$ for control. We present analysis and simulation studies which demonstrate the ability of a POPS network to support large scale parallel processing (1024 nodes) using current device and coupler technology.**

I. Introduction

Among the most attractive features of reconfigurable electro-optical interconnection networks is the ability to tradeoff optical channel bandwidth for the wiring complexity of an equivalent electronic implementation. A variety of multiplexing methods have been proposed and implemented that capitalize on this ability. However, in evaluating these methods, both the traditional measures of resource complexity and message latency must be considered as well as two additional constraints which are specific to electro-optical networks. The first constraint is the balanced distribution of optical power throughout the system. The second is the efficient control and routing of messages through the network.

Optical power constraints can limit the size of a system to the number of receiver sites that can be illuminated by a single transmitter. Often these receivers must be illuminated in unpredictable patterns and in the presence of significant losses in the transmission medium. Control efficiency is a measure of the amount of data transferred per control operation. Since in an electro-optic network the bandwidth of the electronic control circuits is typically limited, control bandwidth can become a bottleneck unless each increase in optical bandwidth is accompanied by a corresponding increase in control efficiency.

In this paper, we present an architecture which is a synthesis of a topological solution to the resource complexity and power distribution problems, and a distributed control paradigm which exploits locality characteristics in the message traffic to amortize control overhead over multiple messages. The topology is based on a multiple passive star organization which we call a partitioned optical passive star or POPS network [10]. The control solution is based on the state sequence routing paradigm [5], [6].[1]

Our presentation is organized as follows. Section II describes the background and previous research which motivates this effort. Section III introduces the POPS network topology, and Section IV describes state sequence routing. Section V characterizes network performance by a static analysis of the service time for random message sets. Section VI outlines the distributed implementation of state sequence routing, and Section VII presents simulation data on dynamic performance. Section VIII gives our conclusions and directions for future research.

II. Background and Motivation

Multiple passive star networks are attractive for multiple access single-hop interconnection networks because they offer a maximum connectivity with a constant power budget [1], and are simple, relatively low cost yet robust structures [2]. Using multiple passive stars, it is possible to design completely reconfigurable networks without the use of active photonic switches [11]. Historically, network configurations using passive star components have been implemented using one of three multiplexing technologies [17]: wavelength division multiplexing (WDM), time division multiplexing (TDM), or code division multiplexing (CDM). In general, to be a completely nonblocking network the number of available slots (wavelength, time or code) must be greater than or equal to the square of the number of nodes. If not, then contention resolution must be incorporated into the control.

WDM technologies allow nonblocking networks to be constructed using passive star and broadcast-and-select architectures with large numbers of wavelength slots relative to the

Manuscript received August 14, 1995; revised March 6, 1996. This work was supported by the Air Force Office of Scientific Research under Contracts 89-0469 and F49620-93-1-0023DEF.

D. M. Chiarulli, R. P. Melhem, and G. Gravenstreter are with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260 USA.

S. P. Levitan and J. P. Teza are with the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15260 USA.

Publisher Item Identifier S 0733-8724(96)05425-4.

[1] A preliminary version of this paper appeared in [7].

number of nodes. However, application of WDM technology is hampered by the need to use closely spaced wavelengths to obtain the required number of slots when the network is scaled to more than a trivial number of processors. This technology is currently capable of wavelength separations on the order of 1 nm yielding a total on the order of 100 slots given a typical fiber transmission window with constraints on the total power of all wavelengths [3]. In addition, it is necessary to use rapidly tunable laser sources, rapidly tunable filter receivers, or both, to obtain sufficient interconnectivity and throughput. Only if both tunable receivers and tunable transmitters are used is it possible to use a number of wavelengths less than the number of interconnected nodes. Currently, the tuning speed for tunable lasers is on the order of several nanoseconds [15] and for tunable filters is on the order of several microseconds [21]. Conflicts in WDM systems consist of wavelength conflicts in which two or more messages are routed using the same wavelength, and destination conflicts in which a message may be routed on a wavelength to which a receiving node is not tuned. These conflicts can be resolved with varying success using any of several distributed control protocols [16]. However, the need to use a control channel may place a limit on the network capacity imposed by the bandwidth of the control channel [9].

CDM requires that data bits be encoded with respect to the address of the receiver using a sequence of bits at a much higher rate than the data bit rate [20], [23]. This requires high speed signal processing be performed at both the transmitting and receiving nodes in order to obtain usable data rates. Several implementations using optical processing have been proposed [8], [13]. In addition, a set of orthogonal codes is required to uniquely address each receiver node. This limits the scale of the network to the length of the code and to the speed of the signal processing used for the encoding and decoding.

TDM systems assign fixed time slots to specific paths in order to obtain complete connectivity. For burst type traffic, both throughput and latency are relatively poor since the utilization of the slots is low and the wait time between paths is fixed. In many implementations, throughput may be sacrificed in favor of simple control. Other designs have implemented adaptive scheduling to increase channel utilization and improve performance [4].

In considering the combined issues of topology and control, our investigation has specifically focused on the application of optical technology which can be implemented currently or in the near term. Therefore, our designs are driven by three constraints. First, for performance reasons, a totally connected single-hop network is necessary, i.e., a connection between any two nodes must not require forwarding by another node. Second, power distribution must be uniform, and the power budget for any path must not directly limit the number of nodes in the network. Third, designs must be implementable with "off the shelf" components such as fixed wavelength transmitters and receivers and couplers of nominal fanin and fanout.

Our solution for meeting these constraints uses a multiple passive star topology with control implemented in the time domain. Multiple passive star couplers can be configured to provide the connectivity required by the first constraint within the power budget limitations imposed by the second. By controlling the network in the time domain, reconfiguration switching can be implemented electronically by a selection operation at the transmit and receive sites. This provides for faster switching times than are currently available in either WDM or spatial electro-optic switches.

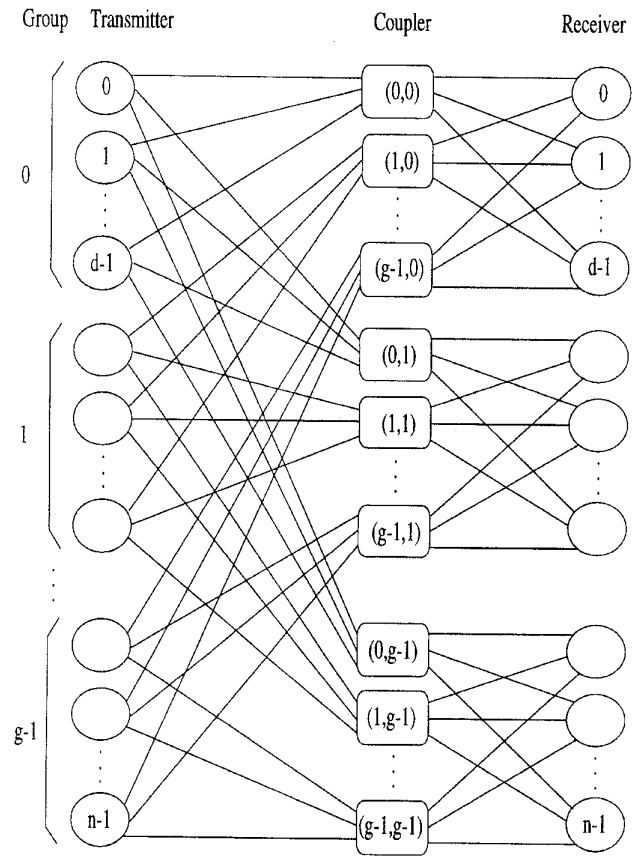

Fig. 1. POPS network topology.

III. Network Topology

In this section, we describe the topology of POPS networks as shown in Fig. 1. POPS networks use a multiple passive star topology in which the interconnected nodes are partitioned into groups such that each group shares common inputs or common outputs from among a set of passive star couplers. POPS networks are distinguished from other types of multiple passive star topologies [1], [11] as follows. First, all couplers have equal fanout and are symmetric in the degree of fanin and fanout. Second, the nodes are completely connected with couplers arranged in parallel and without hierarchical interconnections. Thus, a path exists between every pair of nodes and each path traverses exactly one coupler.

All POPS networks are characterized by the parameter triple (n, d, r). The first parameter n is the number of nodes. In Fig. 1, each of the nodes to the left and to the right of the couplers is actually the transmit and receive logic of the same

node. The second parameter d is the partition size. This parameter sets the size of each group and the fanin/fanout of the couplers. The third parameter r characterizes the redundancy of the network. Redundancy refers is the number of paths available between any pair of nodes. These paths may be independently switchable or may be parallel paths for fault tolerance or bit parallelism. The implications of redundancy in POPS networks is a focus of ongoing research. In this paper, we restrict ourselves to networks with a redundancy of one.

It is useful to additionally define a fourth parameter $g \equiv n/d$ which represents the number of groups into which the nodes have been partitioned. Each of the couplers in Fig. 1 is identified by a double (i, j) where i is the group number of the nodes which share the input side of the coupler and j is the group number of the nodes which share the output side of the coupler. A POPS network is constructed by appropriately connecting couplers for all possible values $(0 \leq i < g, 0 \leq j < g)$. Each node is connected to the inputs of g couplers and is capable of independently transmitting a message into any one. Similarly, each node has g receivers connected respectively to the output side of g couplers and may independently receive a message from any one of the couplers on its receive side. Switching and configuration of the network is accomplished by selecting the appropriate output and input channels at each node. Thus, a message which traverses a coupler (i, j) is routed by a pair of selection operations, one at the transmitter and one at the receiver. The transmitter selects the coupler corresponding to the receiving node group j and the receiver selects the coupler corresponding to the node group i of the transmitter.

From this discussion, it is clear that the size of any POPS network is the product of two parameters, $n = dg$. The choice of these two parameters determines the number and distribution of resources in the network. Specifically, the size of each partition d determines the fanout requirement for each coupler. The number of groups g determines the number of transmitter/receiver channels per node as well as the total number of couplers in the system.

The complexity growth of the network under scaling also depends upon the choices made for d and g. We refer to a specific relationship between these parameters as a *scaling rule*. A scaling rule determines how additional nodes are allocated between groups when the network in increased in size. A *fixed-g rule* keeps the number of groups constant and increases the size of the network by increasing d. Under a fixed-g rule, network resource counts for couplers remains constant. Only the degree of fanout for the couplers is increased. The total number of transmitters and receivers in the network increases linearly with n. Conversely, the network can be scaled using a *fixed-d rule*. Under this rule, the node count is increased by adding additional groups and keeping the size of each group constant. In this case, coupler fanout remains constant and the total number of couplers increases as g^2. The total number of transmitters and receivers in the network increases as ng. The total number of messages that can be carried simultaneously by the network also increases as g^2.

One important advantage of POPS networks over other partitioned networks, is their ability to scale up using a trade off between the two parameters d and g. As an example consider another scaling rule, a *square-root rule*, in which d and g are not considered independently, but instead are increased according to the relationship $d = c\sqrt{n}$ where c is a proportionality constant. Under this rule, the total number of couplers in the system increase linearly in proportion to n and the total number of transmitter/receiver channels in the system increase proportionately to $n\sqrt{n}$.

Given these morphological characteristics and current technological limits on coupler fanout [19], it is currently feasible to construct POPS networks on the order of a few thousands of processors. For example, using a coupler fanout of 64, a 1024 node system would have 16 groups, 16 transmitter and receiver channels per node and 256 couplers. Further, if one were to adopt integrated optical [18] or free space coupling [12] in lieu of fused fiber, even higher fanouts and larger scale systems would be possible. The free space domain has a very interesting potential for two reasons. First, as optical bandwidths continue to increase more rapidly than the data bandwidth of each processor, high fanout coupling is more desirable. Second, in free space the essential communication resource is total spatial bandwidth rather than the number of physical coupler devices.

The performance of various POPS topologies is the subject of later sections of this paper. We turn now to a discussion of a control system appropriate for these networks.

IV. State Sequence Control

In addition to a scalable topology, another issue for reconfigurable interconnection networks is the design of a low latency control system. In our implementation of the POPS network messages are routed using a technique called *state sequence routing* [5], [6]. The goal of state sequence routing is to decouple the bandwidth of the optical channels in an electro-optically switched network from the bandwidth of the electronic control system which routes each message. In other words, state sequence routing provides a mechanism where network throughput is not directly constrained by the limited bandwidth of the control system.

Consider a POPS network interconnecting a set N of n nodes. Within this network, an end-to-end connection between any two nodes is referred to as a path p_{io} such that $i, o \in N$ and $p_{io} \in P$ where the set P represents the complete connection space of the n^2 possible paths. If the network is partitioned into g groups, then the network can establish at any one time up to g^2 of the paths in the set P. This is accomplished by programming each of the source and destination nodes to respectively select specific output and input channels. That set of channel selections is collectively referred to as the network state and the corresponding set of paths implemented is referred to as a *mapping*.

Because individual pairs of paths may block over contention for either a specific coupler or a destination, not all sets of paths may constitute a mapping. Let M be the set of all mappings for a specific topology. If T is any arbitrary set of paths which corresponds to the current traffic, then it is always possible to partition T into subsets, t_i, such

that each of the subsets is a mapping, $t_i \in M$, and $T = t_1 \cup t_2 \cup t_3 \cup \cdots \cup t_k$. In other words, there exists a sequence of mappings which contains all paths in the current traffic and a corresponding *state sequence* which implements those mappings. The *sequence length* k is the number of mappings required to route all of the paths in the current traffic.

In the simplest implementation of state sequence routing, the set of paths T is known *a priori* because it represents the static embedding of a fixed computational structure. For example, state sequences can be derived for a set of paths which represent the links in a mesh, a tree, or a cube connection structure. This sequence is repetitively applied to the switches in the interconnection network. A node may transmit whenever it detects that the network state implements a mapping which contains the desired path. If no such path exists in the current mapping, the node must wait. In the extreme, a state sequence can be devised to implement all of the n^2 paths in a completely connected network [24]. In this case, state sequence routing is identical to time multiplexing. Unfortunately, the latencies inherent in a fully time multiplexed implementation, as well as for most other static embeddings, is prohibitive. This is because, although the computational structure is regular, and the corresponding paths are repetitively available, the channel utilization is low and a large percentage of the available bandwidth is wasted.

In addition to the latency argument, another problem with static sequences is that for the majority of computing problems the computational structure cannot be known *a priori*. The current traffic set T changes dynamically as the computation progresses. Therefore, since the state sequence is applied repetitively to the switching elements in the network, the control problem becomes a matter of transforming the state sequence to track the dynamic changes in the traffic. The essential point is that the control unit needs only to respond to the changes in the traffic and is not required to respond to individual messages. Since communication patterns in a multiprocessor environment tend to exhibit locality characteristics [14], the rate at which changes occur in the traffic is significantly lower than the message generation rate. Thus, we can decouple the performance of the control system from the throughput of the network.

There are several methods of implementing dynamic sequence transformation. They can generally be classified by two characteristics, those which use fixed or variable sequence lengths, and those which allow or do not allow preemption of paths in the current sequence. For this paper, we focus on a method which uses fixed length sequences and in which path preemption is allowed. Under these assumptions, the selection of the path which must be preempted is the essential function of the control algorithm.

To understand this algorithm, it is useful to examine the correlation between this form of state sequence routing and the virtual memory page replacement problem. A virtual memory system attempts to emulate the functionality of a large address space by using a small amount of physical memory. Similarly, we are emulating the n^2 paths of a fully connected network using a small number of physical channels. By this analogy, a path is analogous to a memory location and a mapping is analogous to a page frame. The state sequence, which is a set of mappings including all of the currently available paths, is analogous to physical memory, which is a set of page frames containing all of the currently accessible memory locations.

In a virtual memory system, when a memory fault occurs, the memory controller replaces one of the pages in physical memory with a new page containing the required location. In state sequence routing, a sequence fault occurs when none of the mappings in the current state sequence contain a required path. In response to a sequence fault, the controller must replace one of the mappings in the sequence with a mapping that includes the required path. For both control tasks, the most significant problem is selecting the entity (page or mapping) that must be preempted to make room for the replacement. Optimally, the algorithm should choose to remove the entity which will be used at the most distant time in the future.

Since future behavior cannot be precisely predicted, an optimal algorithm cannot be devised. However, the literature on operating systems contains numerous algorithms of varying complexity for selecting a candidate for replacement [22]. As discussed below a central focus of our research is the evaluation of various replacement algorithms to state sequence routing. Although the replacement algorithm is important for reducing both the number of faults, and fault service time, message latency is also tied to the choice of sequence length. In the next section, we turn to this issue.

V. Static Performance Analysis

In a POPS network using fixed length/preemptive state sequence control, the average control latency, defined as the time between the generation of a message by a processor and the launching of that message into the network, is given by

$$L_{\text{avg}} = \left(\frac{k \cdot p}{2}\right)(1-h) + (f + k \cdot p)h$$

where k is the sequence length, p is the time of a sequence step, h is the fault probability, and f is the service time for a sequence fault. In other words, messages for which a path exists in the sequence must on average wait for one half of the sequence period, $k \cdot p$. When a sequence fault occurs, a message must wait for one sequence period to determine that the path is not in the sequence, plus the fault service time.

As a rule of thumb, for systems like these, the maximum fault probability that can be tolerated is related to the fault service time (i.e., $h \cdot f \leq const$). However, in this case there is a substantial contribution to the latency of faulting messages from the $k \cdot p$ terms. In fact, for many implementations the fault service time will be significantly less than the sequence period. Therefore, average message latency is critically dependent on the ability to efficiently deliver large volumes of traffic using a minimal sequence length.

One of our reasons for choosing the POPS topology is that it can deliver messages very efficiently. We show this with two static analyzes of random traffic sets T, to determine the number of steps in a sequence which would deliver the entire set without repetition. In the first analysis, a (1024,128,1) POPS network was considered. This network consisted of

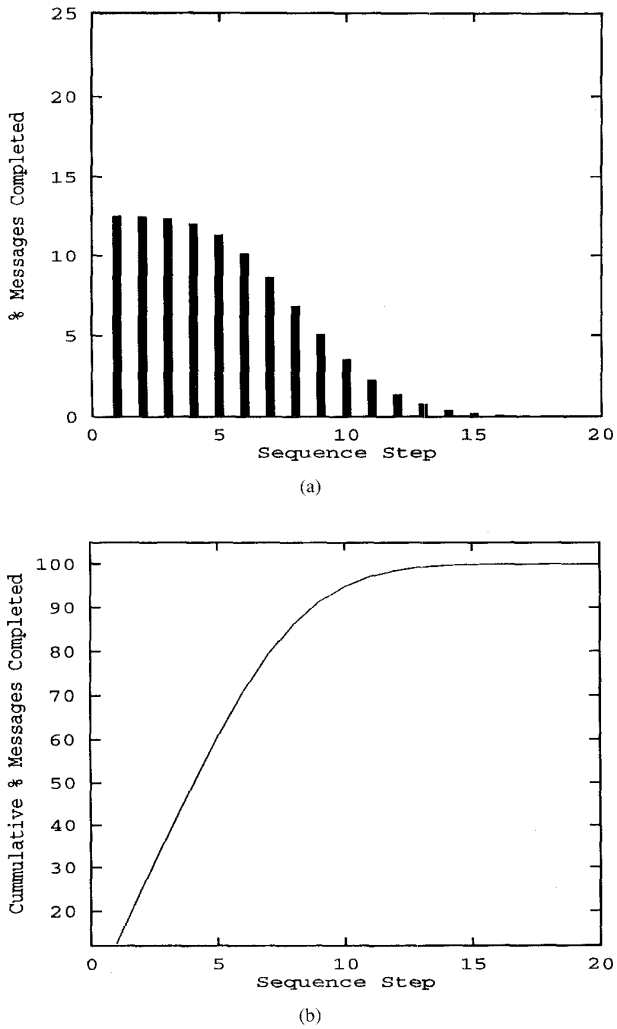

Fig. 2. (a) The % messages completed versus sequence step. (b) Cumulative % of messages completed versus sequence step.

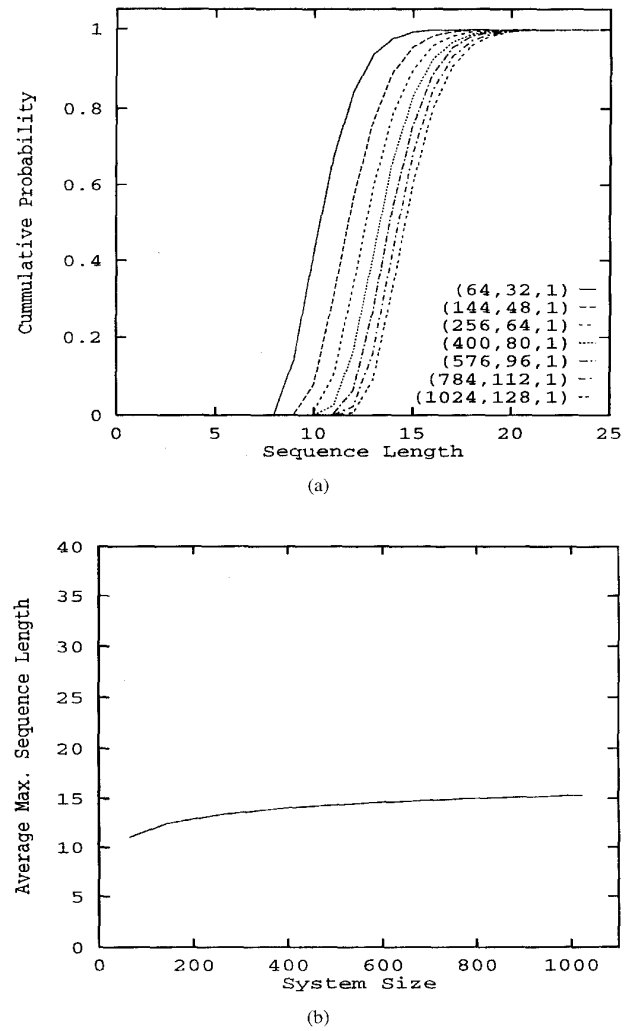

Fig. 3. (a) Complete set transmission probability versus sequence length. (b) Complete transmission sequence length versus system size.

1024 nodes in eight groups, and used a total of 64 couplers. Ten thousand traffic sets, each set consisting of 512 messages, which corresponds to 50% active processors, were generated. The results shown in Fig. 2(a) are the average percentage of each traffic set which was transmitted at each step. Fig. 2(b) shows the cumulative percentage of messages delivered on average after each step. From the latter result, we see that over 94% of the messages were delivered after only 10 steps and 100% of the messages were delivered in 22 steps. Further, Fig. 2(a) shows that in the first five steps, between 12% and 12.5% of the message set of 512 messages were transmitted per step. This corresponds to a channel utilization of nearly 100% for the initial steps in the sequence.

In a second analysis, we considered several different network configurations ranging in size from $n = 64$ nodes to $n = 1024$ nodes, with group sizes given by the scaling rule $d = 4\sqrt{n}$. By adopting this scaling rule, each increase in the number of nodes was accompanied by a linear increase in the total number of couplers, and an increase in the number of transmitter and receiver channels proportional to $n\sqrt{n}$. For each configuration, an analysis was made of the maximum number of steps required to deliver a random traffic set with size corresponding to 50% active processors. Fig. 3(a) is a plot of the probability that a random traffic set can be delivered by a specific length sequence without repetition. Each curve corresponds to one system size over the range of the analysis. The scaling characteristics are summarized in Fig. 3(b). This data shows that the average sequence length does not significantly increase even when the system is scaled over two orders of magnitude.

These results show that the number of steps required to deliver a specific static traffic set is relatively short and that this is consistently the case even for large scale systems when the square root scaling rule is observed.

For dynamic traffic, an underlying assumption is that changes in the traffic set will occur slowly due to locality. Our repetitive application of a fixed length sequence is based on that assumption. From the latency expression above, we

determined that the performance of a dynamic system is critically dependant on the choice of sequence length k. The static results show that sequences with small k can deliver substantial percentages of the current traffic set. With dynamic changes, transformations of the sequence are also required at a rate given by the sequence fault probability h. The minimum value of h is determined by the locality in the traffic. However, the actual value may be greater, depending on contention for slots within the sequence as paths are preempted by the transformation algorithm. Such preemptions are control overhead and should be kept minimal. The rate at which they occur depends on the sequence length and on how nearly the transformation algorithm can achieve optimal replacement. The static results suggest that small values of k will provide sufficient slots to deliver the messages with minimal contention. Thus, fault probability as a function of k can be expected to decline sharply as k is increased to approach the values suggested by the static analysis. As h declines, our rule of thumb relating the service time f to fault probability allows us in turn to implement slower but more optimal replacement algorithms. Better replacement algorithms, in turn, should further reduce the fault probability. The simulation data for the performance of a dynamic system presented in Section VII confirms this relationship.

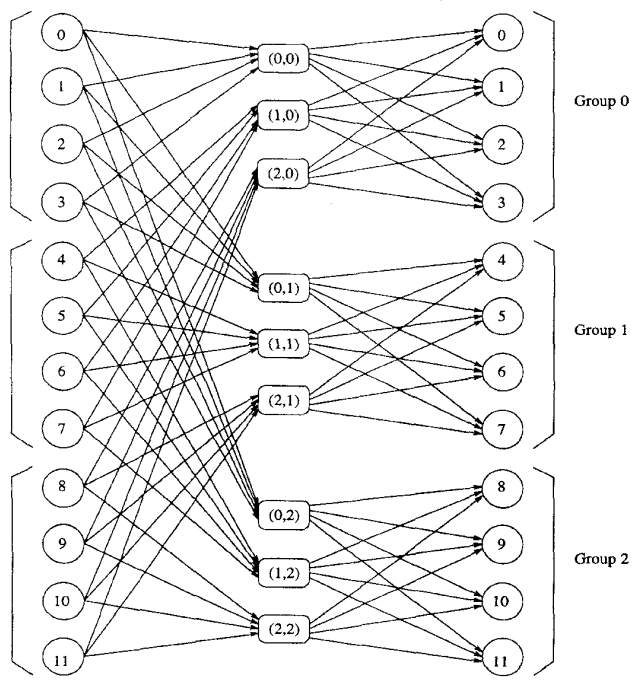

Fig. 4. The (12, 4, 1) POPS network.

VI. Distributed State Sequence Control

We turn now to the implementation of state sequence control for a POPS network. There are two functions which the control system must provide for the network. The first is state sequence generation. This part of the control system keeps track of the current state sequence and transmits, in turn, each state word in the sequence to the control circuitry in the nodes. The sequence is transmitted repetitively and message traffic enters the network when the current state word corresponds to a mapping which contains the path required. The second function is state sequence transformation. This part of the control system is responsible for monitoring the nodes for sequence faults. When a sequence fault occurs, the sequence transformer modifies the state sequence to provide the requested path.

To remove the bottleneck which would occur with a centralized controller, both the state sequence generation and state sequence transformation functions are distributed to designated nodes within the network. The control problem is partitioned the same way that the POPS topology partitions the nodes. Each group of nodes in the POPS topology has within it a designated control node. Since each state word corresponds to a mapping, the state sequence generation function is partitioned such that the control node for each group generates a portion of the state word corresponding to any path which originates in that group. Similarly, state sequence transformation is partitioned such that the control node in each group services sequence faults for paths originating in that group.

A consequence of this partitioning is that contention among the transmitters of a group for the coupler inputs, and contention among the coupler outputs for the receiver channels at each node must be resolved independently. For this reason, the state sequence generation function is divided into a two phase control pipeline. The first phase of the control pipeline resolves contention for the couplers and the second phase resolves contention for the receiving nodes. One advantage to this system is that it allows the use of the second phase to communicate sequence faults from each node to its respective controller.

To illustrate distributed state sequence control, consider the (12, 4, 1) POPS network shown in Fig. 4. It consists of 12 nodes numbered 0 to 11, in three groups of 4 nodes numbered 0, 1, and 2. The group number of any node is the node number div 4. Nodes 0, 4, and 8 are designated as control nodes for groups 0, 1, and 2, respectively. As in the previous example, couplers are identified by the double (i, j) where i indicates the upstream group whose transmitters fanin to the coupler and j the downstream group whose receivers are fanned out from the coupler. Thus, any path which originates from a node s and ends at node d will pass through coupler $(i = s \operatorname{div} 4, j = d \operatorname{div} 4)$.

In the first phase of the control pipeline, the control node for the ith group must generate a state word consisting of a set of fields $T_{ij}R_{ij}$ for all possible node groups j where T_{ij} is the node designated to transmit into coupler (i, j) and R_{ij} is the node designated to receive the output from coupler (i, j). As shown in Fig. 5(a), in phase one the control nodes simultaneously transmit this word into all couplers connected to the group. Each node in the network simultaneously receives and buffers the control word from all controllers in the network.

Therefore, at the end of the first control phase, each node is aware of all paths in the current state word. Based on the state of its transmit and receive buffers, it may modify the

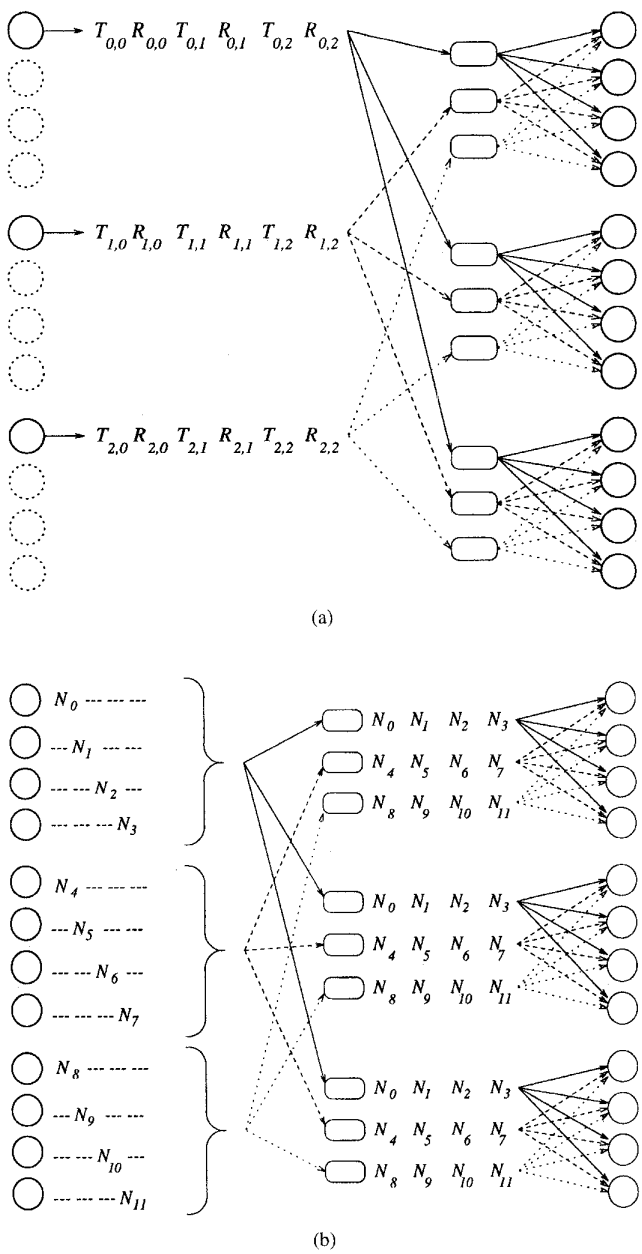

Fig. 5. (a) Control pipeline phase 1. (b) Control pipeline phase 2.

sequence during phase two to reflect its current activity. For example, a node may see a path in the sequence originating from itself, but it may not have a message in its output buffer. Similarly, a node may see a path to itself but not have an input buffer available in the receiver for the group, or it may discover that multiple paths end at one of its receivers. Based on these conditions, each node i generates one field N_i of the modified phase two-state word. N_i encodes the combination of the current transmit and receive activity for the node. Possible values for each are shown in Table I.

Each node transmits its portion of the modified state word simultaneously on all of its output channels. Within a group, each node field is positioned in the output word based on the node number offset within the group. All other bits in the output word of each node are zero. As shown in Fig. 5(b), one segment of the phase two state word is produced for each group by an optical OR operation within the couplers connected to that group. Combined in this fashion, all of the phase two state word segments are simultaneously received and buffered by each node. Once the modified state sequence is received by the nodes, each node which has data in its output buffer examines the modified sequence fields for the source and destination to determine if the required path exists. The path exists if the node states for both the source and destination nodes indicate transmit and receive activity corresponding to the required path. When this condition is met, the message is transmitted. Since the modified state word also indicates fault conditions, each of the control nodes examines the segment generated within its own group for any node state which indicates a sequence fault. If so, the sequence fault service algorithm selects a location in the phase one state sequence into which the faulting path is placed.

TABLE I
TRANSMIT AND RECEIVE NODE CONDITIONS FOR PHASE 2

Transmit Condition	Description	Receive Condition	Description
Idle	Transmit buffer empty or no path in mapping.	Idle	No path in mapping terminating at node
Transmit n	Message to destination n, mapping unmodified	Busy	Path in mapping but no input buffers available.
Fault n	Sequence fault for dest. n, sent in lieu of idle	Receive g	Path in mapping from group g, mapping unmodified

An important issue relative to both performance and scalability of this method is the encoding method for information in the control pipeline. For phase one, $T_{i,j}$ and $R_{i,j}$ encode a node offset within each group. Thus, the size of each field of a state word segment in phase one is $\log_2(d)$ bits long and each segment is $g \log_2(d)$ bits. In phase two, each of the N_i fields must encode the combined transmit and receive activities of a node. From Table I, there are a total of $2n+1$ transmit states and $g+2$ receive states. Therefore, the field size required to encode each node activity is $O[\log_2(n \cdot g)]$ and the total length of each modified state word segment is $O[d \log_2(n \cdot g)]$ bits.

The encoding complexity of the control word is important because it represents the amount of information which must traverse the network within each sequence step. In the latency equation in Section V, the period of a sequence step is represented by p and contributes to the latency in the same relationship as the sequence length. The exact rate of growth in p relative to the size of the network is implementation dependent. However, if we assume that the control phases are pipelined, the maximum number of bits per segment are in phase two. Thus, for systems which are scaled using the square root scaling rule, p grows as $O(\sqrt{n} \log_2(n\sqrt{n}))$.

VII. SIMULATION RESULTS

The analysis in Section V demonstrated that for static traffic sets it was possible to deliver the entire message set in a relatively small number of steps. It was also shown that

the number of steps required was essentially constant when systems were scaled using the square root scaling rule. For fixed length sequences and dynamic traffic sets, this result is useful to guide us in our selection of the sequence length k, but provides little insight as to the relationship between k and h, the fault probability. In this section, we address this issue with a set of simulations to characterize the fault probability and latency in terms of sequence length in a network under dynamic control.

The simulation model is a POPS network with user definable parameters for system size, partition size, sequence length, and traffic profile. The topology is the one shown in Fig. 1 and the control system is based on the description in Section VI. The simulator is event driven with its timebase defined to be one step in the state sequence. Thus, each tick includes one step in each phase of the control pipeline and one set of message transmissions and receptions. The propagation latency of the fiber in the network is assumed to be two ticks, and this latency is applied to both control segments and messages. Receive and transmit portions of the state word are skewed to take this latency into account. Each node is assumed to have one output buffer and g input buffers, one per group, for messages. Thus, only one message at a time is available for output from a transmitting node. Each receiving node is capable of independently accepting messages from different source groups, one per channel. However, each receiving channel is considered busy for a minimum of two ticks after a message is received to allow time for the node to consume the message. If multiple messages are received, the busy time is extended proportionately to account for the contention on the node's processor system interface. Message latency is defined to be an interval starting when a message is placed in the output buffer of the source node and ending when the message arrives in the input buffer of the destination node.

Message traffic for each simulation run consists of synthetically generated bursts of messages from each processor. All messages in each burst share a common source and destination and the particular destination node is generated at random for each burst at each source node. The length and timing characteristics of each burst are also randomly generated within ranges set by three simulation parameters: *burst length*, which controls the number of messages in each burst, *burst interval*, which controls the number of ticks between bursts, and *burst rate*, which sets the number of ticks between the messages in a burst. Each of these parameters are designed to allow simulation runs to be tailored both to emulate particular traffic loads and specific spatial and temporal locality characteristics.

The spatial locality of a set of messages is defined to be the percentage of messages that are sent from the same source to the same destination as their predecessor. Since each of the messages within a single burst share a common source and destination, only the first message in a burst is "nonlocal." Therefore, b/m is the ratio of the number of bursts b to the total number of messages m which gives the "nonlocal" traffic and $1 - b/m$ is the spatial locality of the traffic. This quantity can be controlled in the simulation by setting the burst length parameter bl since $m = b \cdot bl$.

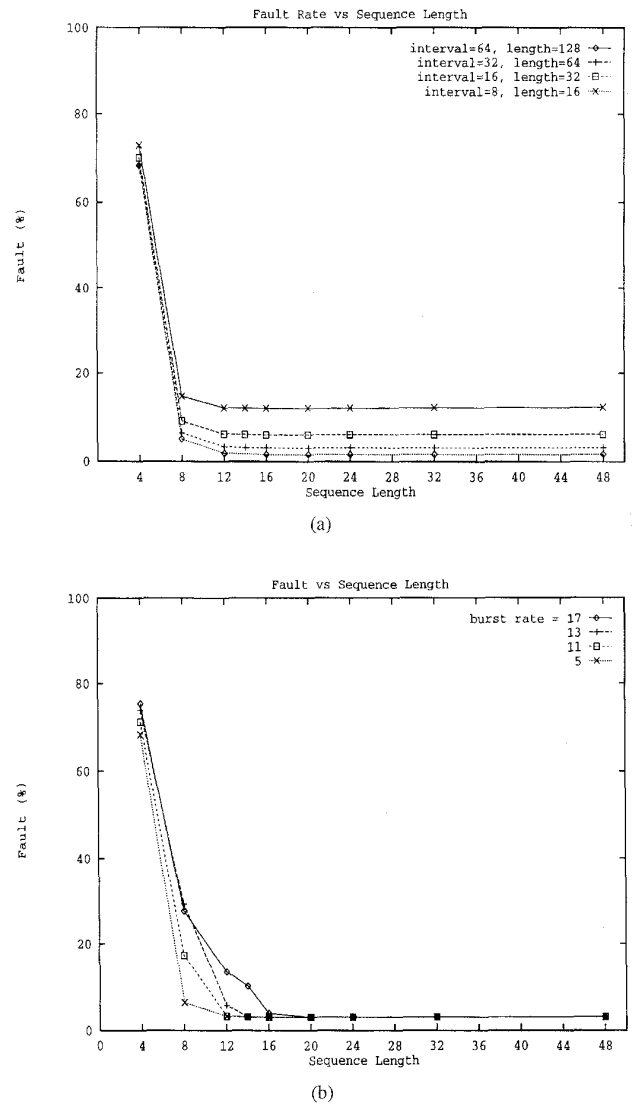

Fig. 6. Fault rate versus sequence length. (a) Efffect of spatial locality. (b) Effect of temporal locality.

Similarly, temporal locality in this context is the inverse of the average delay between successive messages from each processor $1/(br)$ this can be controlled by br the burst rate parameter.

Finally, the overall network *demand* is the average number of new messages per tick generated by all processors as a percentage of the total network capacity. The network capacity, in messages per tick, is simply the number of couplers. n is the number of nodes and bi is the burst interval.

$$\text{demand} = \frac{(\text{messages})/(\text{time})}{(\text{capacity})/(\text{time})} = \frac{n \cdot bl \,/\, (bl \cdot br + bi)}{g^2}.$$

Sequence faults cause path preemptions in the sequence according to a "not used recently" (NUR) replacement algorithm [22]. In this algorithm, each entry in the state sequence is marked as used or unused based on whether a message was actually transmitted when the path was most recently available.

Each controller monitors this information for paths originating in its group using the phase two sequence information. Paths on which no outgoing message occur show an *idle* phase two transmitter state. A path to a busy receiver is considered to be a used path. During replacement, the controller scans the state sequence, one entry per tick, by selecting the state that is next to be issued to the network. If that state was marked as unused during the previous cycle, the new state in substituted in its place. Otherwise, the next state is issued normally. Since each group is connected to g couplers, each controller actually issues g separate paths in each state word. Thus, path replacement proceeds independently and in parallel for each new path within a single controller, as well as in parallel across controllers.

Fault service time f is defined as the interval starting when the node determines that a sequence fault exists to the time at which the new path is available and the message is transmitted. Fault service time includes both a wait time, in which the requested path is buffered prior to service by the replacement hardware and the time required for the replacement hardware to select a path. Thus, fault service time is a function of both the load on each controller, which extends the wait time, and the network load, which increases the number of paths which must be examined. In the former case, the wait time is bounded by d the size of group, for the case where all of the nodes of the source group are in fault to receivers in the same destination group. In the latter case, the wait time is bounded by k the length of a sequence, since after examining k paths and finding them all in use, the algorithm simply preempts the next path. However, the two bounds are not independent. Clearly, if all of the nodes in the group are faulting, then none of the paths in the sequence can be in use. Similarly, if the sequence is heavily used, then only a small number of nodes are faulting.

In the first of our simulation experiments, we selected a (512, 64, 1) network and determined the effect of increasing sequence length on fault rate, network load average, and message latency. Simulation runs were made for sequence lengths ranging from 4–48 and each sequence length was simulated for eight different traffic profiles. Fig. 6 shows the fault rate results. Fig. 6(a) shows four traffic profiles with differing spatial locality. By varying the burst length and the burst interval of the traffic, spatial locality varied from 98 to 87% while demand was kept constant at 140% of capacity, set by a burst rate of 5. The plot in Fig. 6(b) shows a set of traffic profiles which vary the burst rate giving a range of temporal locality from 6–20% and corresponding demands from 45% to 140% of network capacity using a constant burst length of 64 and burst interval of 32.

There are two observations to make about these results. First, for both temporally and spatially local traffic, the number of faults caused by contention for slots in the sequence declines rapidly with increasing sequence length. This is shown in the region of the plot for sequence lengths of $k < 12$. At $k = 12$, the high network utilization demonstrated by the static results in Section V minimizes contention for sequence slots and thus the fault rate approaches a value given by $1 - spatial\ locality$ of the traffic. In other words, one fault for each new burst.

Second, the data for traffic loads with differing temporal

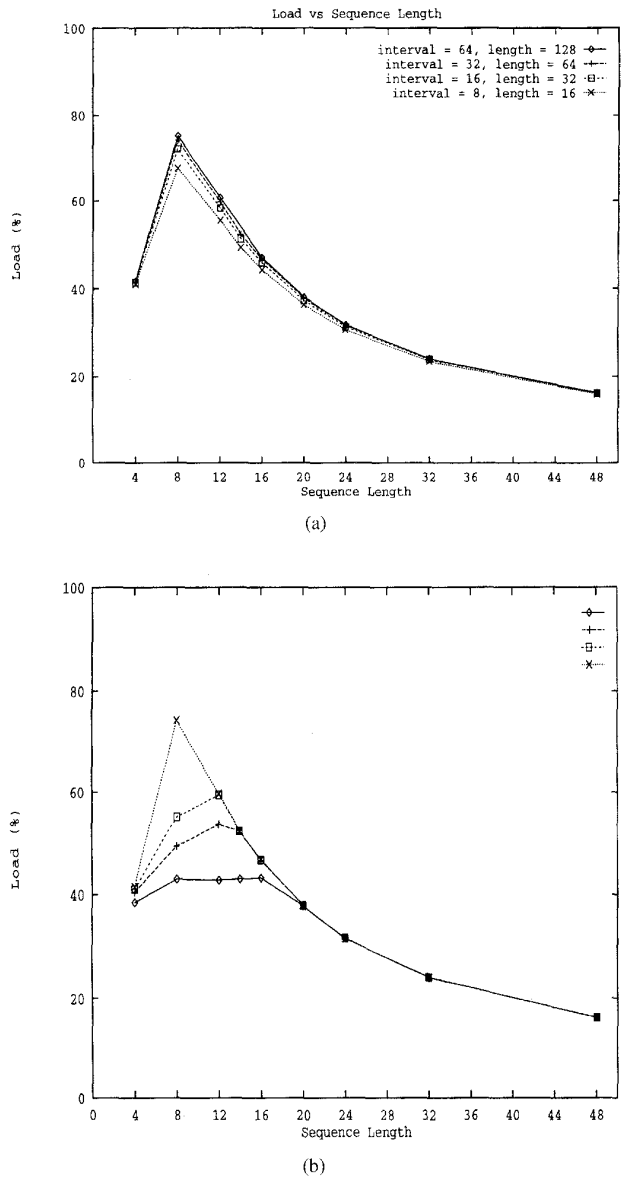

Fig. 7. Network load average versus sequence length. (a) Effect of spatial locality. (b) Effect of temporal locality.

locality in Fig. 6(b) shows what happens as the temporal locality is decreased and network load average is correspondingly decreased. Even for low loads, the fault rate for short sequences shows an excessive number of faults based on sequence contention. More interesting is the fact that the minimum fault rate is approached more slowly than with higher network loads. This is an artifact of the NUR replacement algorithm which marks a path for retention in the sequence each time it is used. Thus, for cases where the sequence length is shorter than the time between successive messages, the replacement algorithm may falsely mark paths as unused and cause preemption. The NUR replacement algorithm implicitly assumes a certain level of temporal locality in the traffic in order to be able to successfully recognize

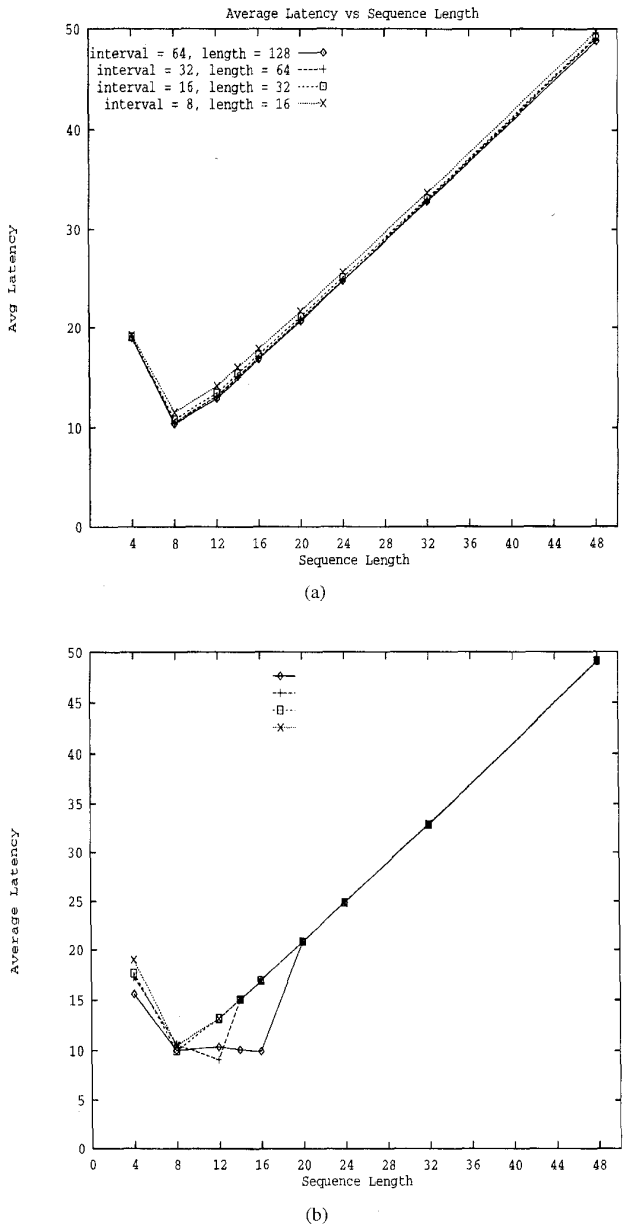

Fig. 8. Average message latency versus sequency length. (a) Effect of spatial locality. (b) Effect of temporal locality.

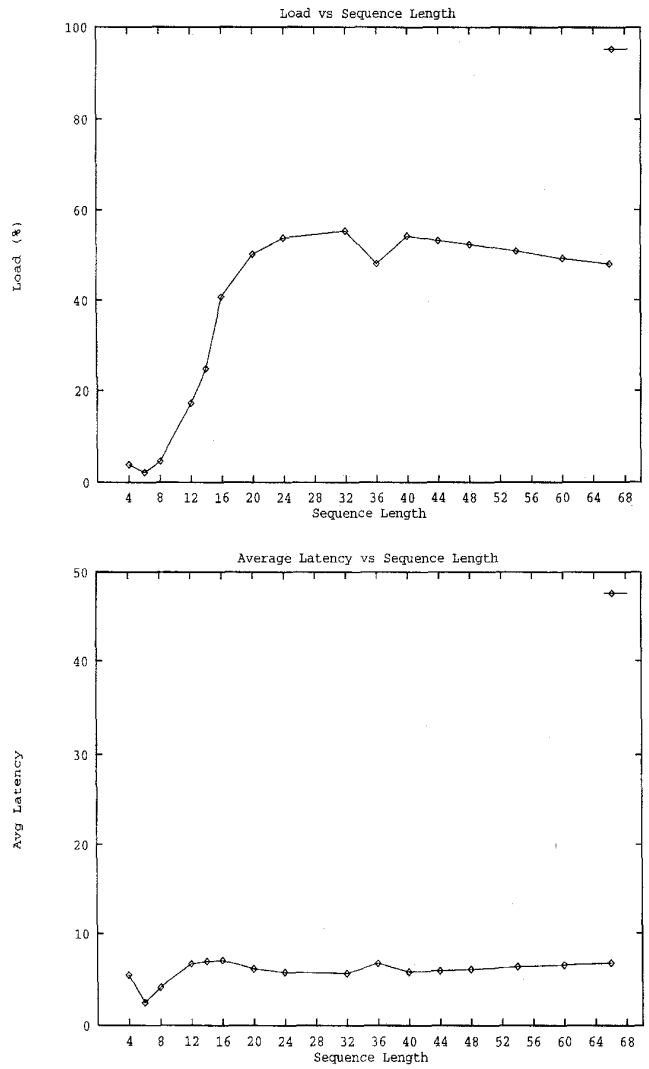

Fig. 9. Network load and message latency versus sequence length using temporal locality data in replacement algorithm.

spatial locality patterns. In a high bandwidth optical network the relatively slow message generation rates of the processors will limit temporal locality. The alternatives are either to make the sequence length longer or to modify the replacement algorithm, we discuss each of these possibilities below.

The simulation data in Figs. 7 and 8 show that increasing the sequence length is not a solution. Fig. 7(a) and (b) are plots of the network delivered load average (or utilization) versus sequence length for the same set of eight traffic profiles used in Fig. 6(a) and (b). For all of the traffic profiles, network utilization increases as the inverse of the fault rate until either the demand is met (e.g., 45% load for the $br = 17$ curve), or until the delay caused by sequence length itself, the $k \cdot p$ term, limits performance. Further, unlike the fault rate, which remains constant as the sequence length continues to increase, the load average for the network decreases following a $1/(k \cdot p)$ curve. Each node is effectively limited to one message transmission per sequence length. Thus, increasing the sequence length, decreases the effective output bandwidth of each node. A similar effect is shown in the results for average message latency in Fig. 8(a) and (b). For short sequence lengths, latency decreases as the number of faults is decreased. However, once the fault rate is minimized, the $k \cdot p/2$ term in the average latency equation dominates and latency increases linearly with k. The additional latency corresponds to the extra wait time as each nodes waits its turn in the sequence.

Our conclusion is that the straightforward NUR replacement algorithm is not appropriate for this network. This is because he intrinsic temporal locality of the processes is lost in the aggregate message traffic, and the NUR algorithm is based on the assumption that spatial and temporal locality are the same.

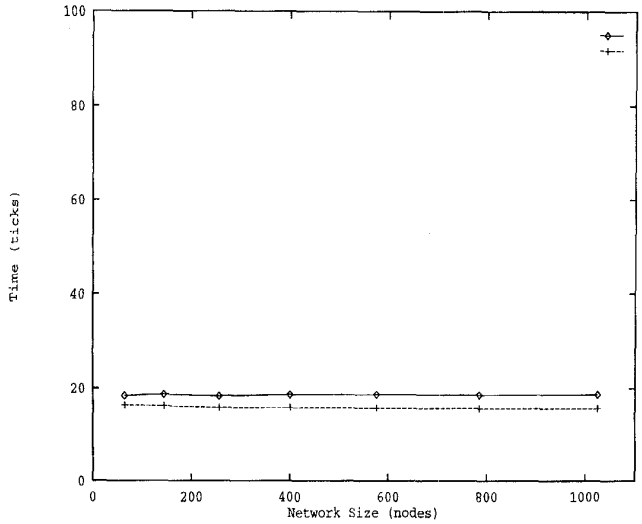

Fig. 10. Latency versus network size.

TABLE II
POPS NETWORK PARAMETERS AND SCALING CHARACTERISTICS

Resource	Notation	Growth
node	n	n
nodes/group	d	$\sqrt{n}$
groups	g	$\sqrt{n}$
couplers	g^2	n
coupler fanout	d	$\sqrt{n}$
transceivers	nd	$n\sqrt{n}$
latency	$(kp/2)(1-h) + (f+3kp/2)h$	kp
control bits	$d \log(ng)$	$\sqrt{n} \log(ng)$
power	d	$\sqrt{n}$

Therefore, we must devise a new replacement algorithm which explicitly takes into account both the spatial and temporal locality characteristics of the traffic. To see how such an algorithm would behave, we performed a set of simulations based on a *hypothetical* replacement algorithm. We assumed that the replacement algorithm has exact information about the temporal locality of the processes. This was accomplished by using the burst rate parameter directly in the replacement calculation, and allowing the replacement algorithm to place multiple copies of the requested path in the state sequence. Each copy was placed at a position separated as nearly as possible by the distance corresponding to the burst rate. However, once a specific region of the sequence was chosen, the exact location of each path to preempt was selected using an NUR algorithm.

The results, shown in Fig. 9 ($br = 5$, $bi = 22$, $bl = 36$), demonstrate that, in contrast to the standard NUR algorithm, both latency and network load are held nearly constant over increasing sequence length. These results are very promising even though they are based on an idealized algorithm and artificial traffic. Handling the more difficult case of actual memory traces, in which the temporal locality must be detected by the replacement algorithm and in which the interval between messages in a burst may vary, is the subject of our ongoing research.

A third set of simulations was conducted to determine the effect on latency when a system scales in the number of nodes. For these simulations, we used the simple NUR algorithm and a relatively low network traffic, to avoid the problems discussed above. Fig. 10 shows plots of the average latency versus network size assuming that the network is scaled using a square root scaling rule. In other words for each network the number of groups is proportional to the square root of the number of nodes. Under this assumption the network complexity, measured in terms of the number of couplers grows linearly. This experiment shows that under these conditions, the average message latency remains essentially constant even when the network size increases over a wide range. The two traces in Fig. 10 show that latency with and without the overhead introduced by the fault service time both remain constant. This is because as the number of groups is increased, so are the are the number of control nodes. As explained earlier, within each controller it is assumed that paths with destinations in different groups are processed in parallel since they are in independent fields of the sequence word. This characteristic of linear scaling with constant latency is highly attractive feature of POPS networks with distributed state sequence control.

VIII. CONCLUSIONS AND FUTURE RESEARCH

In conclusion, we have presented an optical interconnection network which combines the topological advantages of a multiple passive star configuration with the control paradigm of state sequence routing. Table II is a summary of the network parameters and their complexity growth under the square root scaling rule. The significant features of the network include linear scaling in couplers, sublinear scaling in power budget and coupler fanout, and $n\sqrt{n}$ scaling in transceiver count. Also, for the distributed control algorithm, the control information required per sequence step increases sublinearly under scaling. Most importantly, we have shown both by static analysis and by simulation that message latency can be kept nearly constant under scaling for a wide range of system sizes.

The results presented in this paper have demonstrated that in the context of state sequence control, the temporal locality in the traffic must be taken into account by the sequence replacement algorithm. In our current research we are examining various mechanism to estimate the temporal locality of dynamic traffic, as well as the most effective ways to exploit it in new replacement algorithms.

REFERENCES

[1] Y. Birk, "Power-optimal layout of passive, single-hop, fiber-optic interconnection whose capacity increases with the number of stations," in *IEEE INFOCOM'93: 12th Joint Conf. Comput. Commun. Soc.*, 1993.
[2] Y. Birk, N. Linial, and R. Meshulam, "On the uniform-traffic capacity of single-hop interconnections employing shared directional multichannels," *IEEE Trans. Inform. Theory*, vol. 39, pp. 186–191, Jan. 1993.
[3] C. A. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 948–964, Aug. 1990.
[4] W. T. Chen and H. J. Liu, "An adaptive scheduling algorithm for TDM systems," in *IEEE INFOCOM'91*, 1991.
[5] D. M. Chiarulli, S. P. Levitan, R. G. Melhem, and C. Qiao, "Bandwidth as a virtual resource in reconfigurable optical interconnections," in *Optic. Comput. Dig.*, Washington, DC, Optic. Soc. Amer., 1993, vol. 117, pp. 299–302.

[6] ——, "Locality based control algorithms for reconfigurable optical interconnection networks," *Appl. Opt.*, vol. 33, no. 8, pp. 1528–1537, Mar. 10, 1994.
[7] 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 *Proc. First Int. Workshop Massively Parallel Processing Using Optic. Interconnections,* IEEE Comput. Soc. Press, 1994, pp. 70–80.
[8] Y. L. Chang and M. E. Marhic, "Fiber-optic ladder networks for inverse decoding coherent CDMA," *J. Lightwave Technol.*, vol. 10, pp. 1952–1962, Dec. 1992.
[9] P. W. Dowd, "Random access protocols for high-speed interprocessor communication based on an optical passive star topology," *J. Lightwave Technol.*, vol. 9, pp. 799–808, June 1991.
[10] G. Gravenstreter, R. Melhem, D. Chiarulli, S. Levitan, and J. Teza, "The partitioned optical passive stars (POPS) topology," in *Proc. 9th IEEE Int. Parallel Processing Symp.*, Santa Barbara, CA, Apr. 1995, pp. 4–10.
[11] A. Ganz, B. Li, and L. Zenou, "Reconfigurability of multi-star based lightwave LAN's," in *IEEE GLOBECOM'91,* New York, 1991.
[12] H. S. Hinton, "Relationship between the hardware required for photonic switching and optical computing based on free-space digital optics," in *OSA Proc. Photon. Switch.*, Salt Lake City, UT, Optic. Soc. Amer., Mar. 1–3, 1989, pp. 184–191.
[13] A. S. Holmes and R. R. A. Syms, "All-optical CDMA using quasiprime codes," *J. Lightwave Technol.*, vol. 10, pp. 279–286, Feb. 1992.
[14] K. L. Johnson, "The impact of communication locality on large-scale multiprocessor performance," *Comput. Architect. News,* vol. 20, no. 2, pp. 392–402, 1992.
[15] H. Kobrinski, M. P. Vecchi, M. S. Goodman, E. L. Goldstein, T. E. Chapuran, J. M. Cooper, M. Tur, C. E. Zaha, and S. G. Menocal, "Fast wavelength-switching of laser transmitters and amplifiers," *IEEE J. Select. Areas Commun.*, vol. 8, no. 6, pp. 1190–1201, Aug. 1990.
[16] N. Mehravari, "Performance and protocol improvements for very high speed optical fiber local area networks using passive star topology," *J. Lightwave Technol.*, vol. 8, pp. 520–530, Apr. 1990.
[17] B. Mukherjee, "WDM-based local lightwave networks, Part 1: Single-hop systems," *IEEE Network,* pp. 12–27, May 1992.
[18] K. Okamoto, H. Okazaki, Y. Ohmori, and K. Kato, "Fabrication of large scale integrated-optic $n \times n$ star couplers," *IEEE Photon. Technol. Lett.*, vol. 4, pp. 1032–1035, Sept. 1992.
[19] Gould Fiber Optics, *Gould Product Catalog,* 1993.
[20] P. R. Prucnal, M. A. Santoro, and S. K. Sehgal, "Ultrafast all-optical synchronous multiple access fiber networks," *IEEE Select. Areas Commun.*, vol. 4, pp. 1484–1493, Dec. 1986.
[21] D. A. Smith and J. J. Johnson, "Switching speed of integrated acoustically-tunable optical filter," *Electron. Lett.*, vol. 27, no. 23, pp. 2102–2103, Nov. 7, 1991.
[22] A. Silberschatz, J. Peterson, and P. Galvin, *Operating System Concepts.* Reading, MA: Addison Wesley, 1991, 3rd ed.
[23] J. A. Salehi, A. M. Weiner, and J. P. Heritage, "Coherent ultrashort light pulse code division multiple access communication systems," *J. Lightwave Technol.*, vol. 8, pp. 478–491, Mar. 1990.
[24] R. A. Thompson, "The dilated slipped banyan switching architecture for use in an all-optical local-area network," *J. Lightwave Technol.*, vol. 9, pp. 1780–1787, Dec. 1991.

Donald M. Chiarulli received the B.S. degree in physics in 1976, from Louisiana State University, Baton Rouge, the M.S. degree in computer science in 1979 from Virginia Polytechnic Institute, Blacksburg, and the Ph.D. degree in 1986 also from Louisiana State University.

In 1986, he joined the University of Pittsburgh, where he is currently an Associate Professor in the Departments of Computer Science and Electrical Engineering. His research interests are in the area of high-performance computer architecture, specifically optoelectronic systems and interconnect.

Dr. Chiarulli is a member of the Optical Society of America.

Steven P. Levitan (S'83–M'84–SM'95) received the B.S. degree from Case Western Reserve University in 1972 and the M.S. and Ph.D. degrees both in computer science, from the University of Massachusetts, Amherst in 1979 and 1984, respectively.

He is the Wellington C. Carl Associate Professor of Electrical Engineering at the University of Pittsburgh. From 1972 to 1977, he worked for Xylogic Systems designing hardware for computerized text processing systems. From 1984 to 1986, he was an Assistant Professor in the Electrical and Computer Engineering Department at the University of Massachusetts, Amherst. During that time he also worked for Digital Equipment Corporation and Viewlogic Systems as a consultant in HDL simulation and synthesis. In 1987, he joined the Electrical Engineering faculty at the University of Pittsburgh. His research interests include VLSI architectures, optoelectronic computing systems, parallel algorithm design, and HDL simulation and synthesis for VLSI.

Dr. Levitan is on the Editorial Boards of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and *ACM Transactions on Design Automation of Electronic Systems.* He is a member of ACM, OSA, and a senior member of IEEE Computer Society.

Rami P. Melhem was born in Cairo, Egypt, in 1954. He received the B.E. degree in electrical engineering from Cairo University in 1976, the M.A. degree in mathematics, the M.S. degree in computer science from the University of Pittsburgh in 1981, and the Ph.D. degree in computer science from the University of Pittsburgh in 1983.

He is a Professor of computer science at the University of Pittsburgh. Previously, he was an Assistant Professor at Purdue University and an Assistant/Associate Professor at the University of Pittsburgh. He has published numerous papers in the areas of optical interconnections, fault tolerance, systolic architectures, and high-performance computing. He served on program committees of several conferences and workshops and as the general chair for the third International Conference on Massively Parallel Processing Using Optical Interconnections.

Dr. Melhem is on the Editorial Board of the IEEE TRANSACTIONS ON COMPUTERS and was a Guest Editor of the Special Issue on Optical Computing and Interconnection Systems of the *Journal of Parallel and Distributed Computing.* He is a member of the IEEE Computer Society, the Association for Computing Machinery, and the International Society for Optical Engineering.

James P. Teza received the B.S. degree in physics and the M.S. degree in 1995 in electrical engineering from the University of Pittsburgh, Pittsburgh, PA.

He is currently a researcher at the Robotics Institute of Carnegie Mellon University. His research interests include optical interconnection networks for multiprocessor systems.

Mr. Teza is a student member of the IEEE Computer Society.

Greg Gravenstreter received the B.S. degree in mathematics and computer science from the University of Pittsburgh, Pittsburgh, PA, and is currently a graduate student in the Computer Science Department.

From 1971 to 1985, he was a researcher in the Chemical Physics and Computer Science Departments at Westinghouse Electric Corporation's Research and Development Center, Pittsburgh. From 1988 to 1992, he worked as Director of Systems and Operations for Guidance Technologies, also in Pittsburgh. His interests include architectural and systems issues for parallel and distributed systems and massively parallel processing. He is currently a researcher at the Software Engineering Institute of Carnegie Mellon University.