Pre-allocating control bandwidth in an optical interconnection network. *

C. Salisbury † and R. Melhem

Department of Computer Science
University of Pittsburgh

Abstract

To fully exploit the performance of optics in parallel processor interconnection networks, the connections must be entirely optical. This requires the use of circuit switching. Techniques such as time division multiplexing *(TDM) can be used to provide a large number of circuits without the need for program directed control operations. We describe two protocols for dynamically establishing circuits in an interconnection network. We show how TDM can be used to multiplex communication for data and control purposes together in a single optical network. We explore the ability of the protocols and TDM to exploit locality in the communication pattern to improve performance.*

Keywords: *optical networking, time division multiplexing, network control, communication protocols, parallel processing, locality.*

1 Introduction

Future high performance computing needs may be met through the use of clusters of networked computers. Optical technology may be needed to provide the high bandwidth, low latency communication required for these systems. To avoid opto-electronic signal conversions, buffering, and electronic processing delays, the connections through the network should be entirely optical. Such connections are managed through the use of circuit switching techniques.

Circuit switched interconnection architectures often use preallocation techniques for circuit establishment, as those described in [1, 6]. Alternatively, dynamic protocols have been developed to establish circuits in response to the changing needs of parallel and distributed applications. Many dynamic control techniques are based on the use of a broadcast bus, which can be implemented optically using a passive star [5, 8]. However, to interconnect a large number of processors requires the use of a more scalable network architecture. Typically, the architectures which have been proposed are either controlled using packet switching, or do not provide complete all-optical connectivity. Scalable architectures do not provide a broadcast channel for control, and cannot use the dynamic control protocols developed for bus-based architectures [7]. Dynamic control protocols for a multistage interconection network (MIN) have been described in [2, 9]. These approaches rely on the use of a separate electronic network to handle the control protocol.

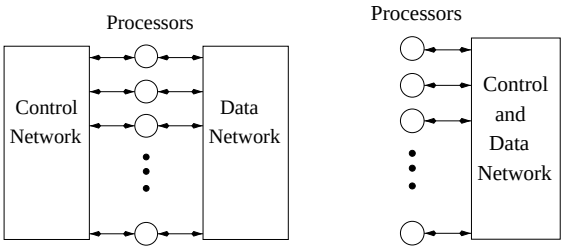
Alternatively, control information and data flow can be provided through a single physical network. One technique for sharing network resources is *time division multiplexing* (TDM) [4, 12]. Using special hardware, a set of circuits is provided by placing the network in a desired state during an assigned time slice long enough to transmit a data packet. The number of netowrk states which are assigned a time slice is called the *multiplexing degree*. A global clock is needed to synchronize the time slices at all processors. A dynamic control protocol that uses TDM to transmit both data and control information over a single network is described for a partitioned optical passive star (POPS) network in [3].

TDM can be used in both optical and electronic networks. It may improve performance when timeslicing a set of network states is more efficient than executing a control protocol to establish new circuits. The concepts of *locality of reference* and *working set* can be applied to the communication patterns of parallel programs, and will affect the performance of a network multiplexed with TDM. The application of dynamic control techniques to the communication patterns found in interprocessor interconnection networks is described in detail in [10].

In the remainder of this paper, we will focus on techniques for dynamic, distributed control where TDM is used to provide circuits for both control and data communication over a single physical network. In Section 2 we introduce three schemes which preallocate different amounts of network bandwidth for control purposes. In Section 3, we

*This work is supported in part by NSF award MIP-9633729 to the University of Pittsburgh

†Current address: Mathematics and Computer Science Division, Argonne National Laboratory, Argonne Illinois



a) Two networks c) One network

Figure 1. Distributed network control

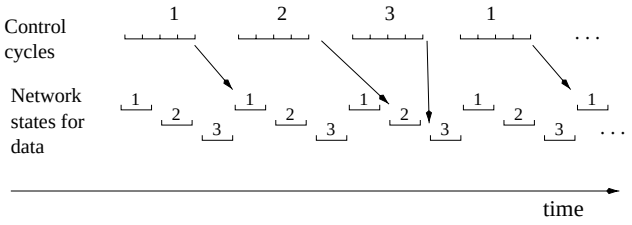


Figure 2. Round robin use of control cycles to build network states. ($K = 3$, $n = 4$)

describe two general approaches to dynamic allocation of network resources based on the use of a control cycle. We examine the performance of these techniques in Section 4, and show how locality of reference can be exploited to improve performance. Our conclusions are in Section 5.

2 Distributed control with TDM

In a circuit switched network, the processing of control messages must take place outside the switching fabric. With a centralized technique, requests are gathered over a control network and a network controller determines the state that will be established in the data network.

Figure 1(a) depicts distributed control with a dedicated control network. Processors exchange control information over the control network. A distributed algorithm is used to allocate network resources for circuits in the data network. Distributed approaches involve multiple steps of communication and processing. Since processors do not have global knowledge, decisions are based on local information and may not be optimal.

The states for control can be multiplexed together with the states for data in a single network using TDM. This is pictured in Figure 1(b). A portion of the bandwidth is allocated for control communication, reducing the bandwidth available for data communication. The allocation also affects the latency of control operations and the rate at which they can be performed.

We consider distributed control techniques where requests are processed in a batch to resolve contention and allocate network resources. The control protocol is executed in a *control cycle* of n steps. Each step has a communication phase to gather or exchange requests, and may be followed by a control information processing phase. The communication pattern is predetermined by the distributed control algorithm, so that each step has an associated network state. The use of a control cycle for processing requests might be appropriate for networks which have globally shared resources. For example, when each processor has a single receiver, the receiver is shared globally. Network architectures could include multi-stage interconnection networks (MINs), optical passive stars, and crossbars.

A straight-forward way of managing a set of multiplexed data states is to build them in a round-robin manner. The relationship between control cycles and network states for data communication is pictured in Figure 2. The steps of a control cycle are shown as being continuous, although this is not necessary. Each control cycle builds a single network state for data communication. These network states can be implemented using time slots in the data network. Circuits are established in the assigned time slot until the data state is rebuilt by a subsequent control cycle.

2.1 An example of distributed control

A dynamic control technique must describe both the pattern of control communication and the method for allocating network resources in response to requests for circuits. Both of these are network specific. To investigate the performance of dynamic control, we developed techniques for controlling a banyan interconnection network for parallel processing systems. A banyan architecture was selected because it is scalable and can be constructed from simple 2×2 optical switches. Unlike a crossbar architecture, all paths through a banyan network pass through the same number of switches. Thus, optical power loss is the same along all paths.

A banyan network interconnecting $N = 2^n$ processors is built with n stages of 2×2 crossbar switches. Each stage has $N/2$ switches. A banyan network provides a unique path between any pair of attached nodes. An example of a banyan network is shown in Figure 3.

As an example of distributed control, we summarize below the control algorithm for a banyan network that was described in detail in [11]. Each step of the control cycle resolves contention for one stage of switches at a time, proceeding from stage 0. Thus, a control cycle requires $\log N$ steps to create a new network state.

Each processor requests a connection by building a control message describing the states of network switches required to form the circuit. The states required to connect processor 0 with processor 5 are shown in Figure 3 ('-' in-

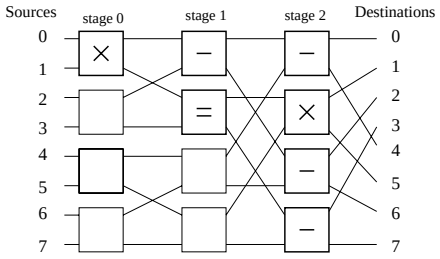


Figure 3. A "reverse cube" banyan network.

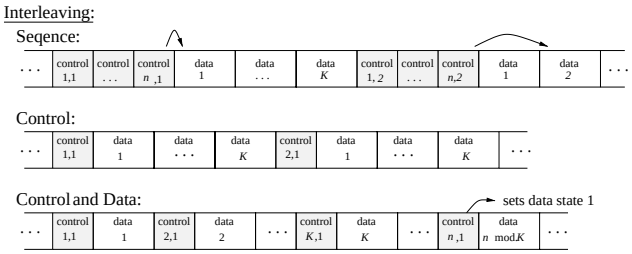


Figure 4. Interleaving data and control.

Interleaving	Max. nr. of packets	Control bandwidth
Sequence	K	$1/(\frac{K}{n}b+1)$
Control	nK	$1/(Kb+1)$
Control and Data	n	$1/(b+1)$

Table 1. Interleaving characteristics.

dicates "don't care"). Since a message from a processor can pass through any of $N-1$ switches, the protocol uses a control message of $\mathbf{O}(N)$ bits.

Each switch in the network can be associated with a subset of processors that could require the switch as part of a circuit. At step i of the control cycle, processors whose addresses differ only in bit position i exchange control information. With this single exchange, each processor obtains information about all possible contention for switches accessible at stage i.

Each processor then resolves contention for network switches by pseudo-randomly selecting one request as the winner, and discarding the losing request. The results are retained for setting the network state. A new control message is then built which combines the requirements of all winning requests, and the procedure is repeated for the next stage of switches. After n steps, a complete network state has been built. Each processor can compare its request to the final state to determine if its request was successful and data can be transmitted in the associated time slot. Requests that fail must be resubmitted into a later control cycle.

2.2 Multiplexing control and data together

The n networks states for control communication must be provided in a predetermined sequence. We assume the K states for data communication are also provided in a sequence. The states in each sequence may or may not be provided in contiguous time slots. We call the use of contiguous slots for both control and data sequences *sequence interleaving*. When data states are not contiguous, every data state must be followed by a control state. We call this *data interleaving*. Similarly, *control interleaving* occurs when control states are not contiguous, and every control state is followed by a data state.

The resulting interleaved sequences are shown in Figure 4. Data interleaving by itself is not shown because it is meaningful only when $K = 1$, and its result is identical to that of sequence interleaving. In the figure, states for control are labeled with the step number within the control cycle and the number of the data state that will be affected. An arrow extending from the n^{th} control state points to the data slot affected by that control cycle.

The duration of a time slot for data communication may differ from the duration of a slot for control communication. The durations are based on the size of a data packet and a control message. Since the resource allocation algorithm must execute between the end of one control communication and the start of the next, the duration of a control slot may need to be extended to provide this time. Alternatively, control interleaving can be used to overlap control processing with data transmission. Let the length of a control time slot be 1 time unit, and the length of a data time slot be b units.

Many performance characteristics of the network can be computed directly from the number and duration of network states for data and control, and the manner in which they are interleaved. These include, for example, the minimum latency to build a network state, the percent of network bandwidth allocated for control, and the number of packets that can be sent over a circuit before the data state will be rebuilt. The latter two values are shown in Table 1. Selection of the interleaving technique represents a tradeoff between the competing needs for network control and data transmission.

3 Dynamic Control Protocols

We consider communication functions to be split between the application program and the processor's interface to the network.

- The program requests the network interface to obtain a circuit to the desired destination. This may be done by a specific network call, or implicit in a SEND.
- The interface constructs a control message and pro-

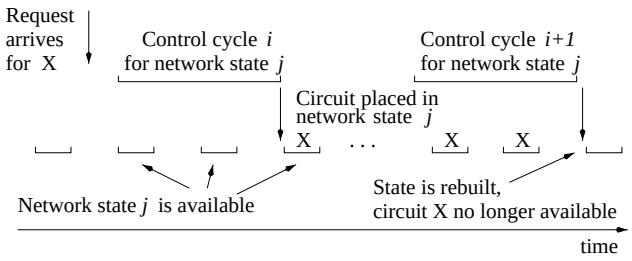


Figure 5. RFE operation.

cesses the request through a control cycle. The control protocol is executed by the interface hardware.

- If the request is granted, messages sent by the application will be transmitted by the interface in the appropriate time slot.
- If the request is denied, the interface can notify the application or it can automatically process it through a subsequent control cycle.

We next describe two classes of dynamic protocol, distinguished by their approach to the reservation of network resources. We then present enhancements that exploit communication locality.

3.1 Reservation with Fixed Expiration

One approach to establishing circuits is to reserve network resources for a fixed duration of time, *e.g.* a fixed number of time slots. We call this protocol Reservation with Fixed Expiration (RFE). When a round-robin scheme is used to manage multiple network states, the time at which each state is rebuilt can be easily computed. At the beginning of each control cycle, all resources are unassigned. The operation of RFE is depicted in Figure 5.

For example, a 2×2 switch is a resource in a banyan network. When allocated during control cycle processing, it changes from an undefined state to either the "straight" or "cross" state. After the entire network state is built, circuits are established by setting the switches in the required position in the appropriate time slot.

There are three consequences of beginning each control cycle with resources in an undefined state. First, a request may be submitted into any control cycle. Second, circuits which are needed longer than provided by the fixed expiration time must be rerequested, even when the communication requirements of the program are not changing. Third, the circuits provided in a network state are determined in a single control cycle. The number of circuits provided reflects the effectiveness of a single execution of the distributed algorithm.

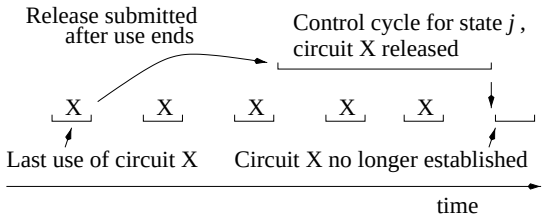


Figure 6. Releasing paths with RER.

When processing a rerequest for an established circuit, contention could cause the request to fail and disrupt the use of the circuit. This could be a problem if timing constraints or bandwidth guarantees are violated. The next protocol does not have this problem.

3.2 Reservation with Explicit Release

The Reservation with Explicit Release (RER) protocol reserves circuits for as long as they are needed. Network resources remain allocated until they are explicitly released by all processors using them. With this approach, control messages are processed as updates to an existing network state. Control cycle processing rejects all requests that would alter the state of reserved network resources. RER can therefore be used to support a higher level protocol that guarantees bandwidth or timing.

With RER, each circuit must be processed through two control cycles: one to reserve the circuit and one to release it. A control message that releases network resources is not subject to contention and always completes successfully. Circuit release is shown in Figure 6. A release control message may be submitted after the last use of a circuit, and network resources are made available for the next control cycle for that state.

Circuit reservation and release

Both circuit reservation and release can be controlled directly by the application. This would allow a circuit to be requested prior to the availability of data in order to overlap control latency with other program activities. However, it is possible for a processor to require a circuit which cannot be accommodated in any of the K multiplexed network states. This represents a potential deadlock situation.

An alternative is to allow the network interface to control circuit establishment and release. A request for a circuit can be generated at the time of a SEND. Similarly, circuits can be released by the interface immediately after the message has been transmitted. This ensures that once a circuit is reserved, the message can be sent, the circuit will be released, and network resources will become available again. Automatic generation of release messages therefore avoids

deadlock and provides complete flexibility in the choice of multiplexing degree.

Network state selection

With RER, a request for a circuit will be unsuccessful if it requires a change in the state of a reserved resource. Processing such a request is unproductive. This can be avoided if the processor retains the network state information developed at the end of each control cycle, and only submits requests which do not require reserved resources to change state. In Section 4, we will show that this restriction can have a significant impact on performance.

In some cases, it may be possible to insert a newly required circuit into any of several network states. In these cases, it may be possible to request the circuit in a state that optimizes performance. For example, the selected state may be the one containing the maximum number of switches already used for other circuits. This may increase the average number of circuits in a network state. Alternatively, a processor may select the state which will be updated the earliest. The optimization algorithm may exploit knowledge of the communication pattern and network topology. For some situations, good optimization algorithms may not be known. However, RER provides the potential for these network optimizations, while RFE does not.

Unlike RFE, the RER protocol allows circuits to be placed into a network state by multiple control cycles. When messages are long enough to reserve circuits over several control cycles, the additional cycles provide opportunities for processors to request circuits that are compatable with the circuits already established. These additional cycles can increase the number of circuits provided in each state, thereby improving performance.

3.3 Protocol enhancements for locality

The basic function of the two protocols cannot exploit repetitive patterns of requests, *i.e.* communication locality, to improve performance. To add this capability, we allow resources which are not reserved to remain in the physical state used during the most recent reservation. Thus, the network state may continue to provide a circuit even after its reservation has expired. Requests for new circuits are compared by the interface to the circuits currently provided by the network. If the circuit is found, data transmission may begin immediately and can safely continue until the network state containing the circuit is rebuilt. This technique should reduce the latency for satisfying some requests by eliminating the need for a control operation.

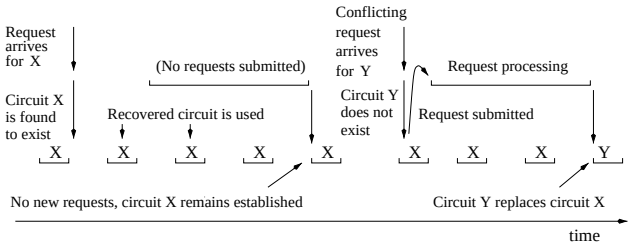


Figure 7. Path recovery in RFE.

Recovery in RFE

We call locality recognition for RFE path *recovery*. A processor looks for a requested circuit in the network state where the circuit was last established. If it still exists, it can be used immediately and can continue to be used until the state is rebuilt without the circuit. Checking only a single state is consistent with keeping the complexity of the RFE protocol to a minimum. The use of recovery is shown in Figure 7.

Discovery in RER

Since an RER request may need to be compared to every network state to find a time slot which can accommodate the new circuit, it is natural to search all network states for the previous existance of a requested circuit. We call this capability path *discovery*. If the entire message can be transmitted before the network state can be changed, it is not necessary to reserve the circuit. If not, a request must be submitted in the appropriate control cycle. It is possible to give priority to these discovered paths during contention resolution, at the cost of increased implementation complexity.

Discovery can be extended further to reduce the use of control cycles by continuing to transmit over the discovered path until either the message is sent or a request from another processor changes the state of a network component in the path. In this latter case, a control cycle is required to reserve the circuit. The advantage is that if the message can be transmitted without reserving network resources, the processing of a release is no longer required. We call this the *don't request* option of discovery. This can provide significant performance gains in parallel processing workloads with repetitive communication patterns, as will be shown in the next section.

4 Performance

The performance of TDM was simulated with the interleaving schemes of Section 2.2 and parallel processing applications using a banyan network.

4.1 Simulation environment

The simulator is written in CSIM and consists of processor components that execute an application script and a network component that simulates the protocol. The application script has a looping structure and specifies a sequence of messages to be generated in each loop iteration and the number of times the loop is to be executed.

While performance was simulated for three communication patterns, space considerations limit our discussion here to only one pattern. The pattern simulates a parallel program in which each processor executes a single loop. In each loop iteration, each processor sends a message to four different destinations. The destinations are chosen randomly at the start of the simulation, and do not change for the duration of the run. The result is a communication pattern with a randomly generated working set. All messages were non-blocking, and processors synchronize at the completion of each loop iteration. Computation and synchronization delays are assumed to overlap communication.

Simulations were also run with a shuffle pattern that could be analyzed manually and a random pattern that did not have a working set. A discussion of these results can be found in [10].

Two message lengths were used. Short messages consisted of a single packet. Long messages were of variable length, with the number of packets taken from a uniform distribution between 25 and 35. Each processor sends 12,000 packets during the simulation.

RER simulations were made using automatic generation of "release" control messages after each data message was sent. In addition, the *don't request* option was used. The length of a control time slot was chosen to be equal to the time required to transmit a control message and did not provide time for control processing. A parallel system of $N = 64$ processors was used, communicating through a reverse cube banyan interconnection network. The network data transfer rate was 1 Gigabit per second, and packet size was set at 50 bytes. Throughput was used as the key measure of network performance. This is reported as the percent of network bandwidth actually used to transmit data.

4.2 Random working set workload

We investigated performance with a range of multiplexing degrees from one to a value sufficient to contain the entire working set. Figures 8 and 9 show the results for the base protocols using short messages. Since the circuit requirements change after every message is sent, the base protocols perform best with a large amount of control bandwidth. For RFE, performance mirrors the allocation of bandwidth for control. RER performs worse than RFE due to release processing.

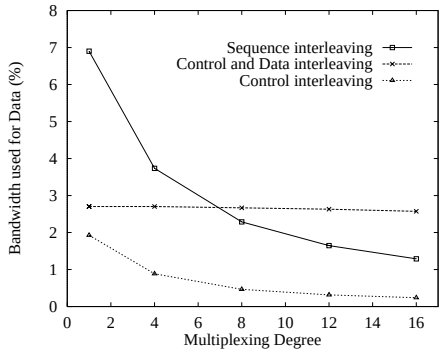


Figure 8. Short messages, RFE base cases.

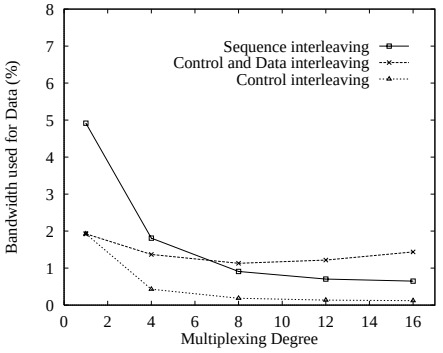


Figure 9. Short messages, RER base cases.

Figures 10 and 11 show the performance with locality recognition. The dramatic improvement at a multiplexing degree of 12 suggests that this degree is capable of containing the entire working set. RFE performance at this degree is still related to the amount of control bandwidth. It requires more control activity to place the working set into 12 network states than it does to place the working set into 16 network states. As the multiplexing degree increases, interleaving to provide additional data bandwidth performs better. The effectiveness of the protocol determines where this transition occurs.

For two interleaving techniques, RER outperforms RFE because it is more effective at grouping requests into non-conflicting subsets. The key reason for this is that the requests submitted into a control cycle for RER do not conflict with the set of circuits currently in the state. Thus, the requests are correlated and the probability of contention between them is less than would be expected from a random set of requests. This increases the portion of requests which can be satisfied during contention resolution. Over many control cycles, the result can be a significant performance improvement even with a multiplexing degree of one, and even when the communication pattern does not have a working set [10].

RER performs poorly with control and data interleaving

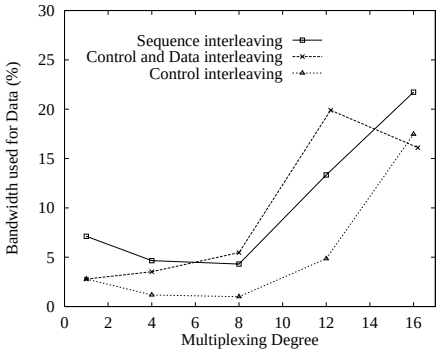


Figure 10. Short messages, RFE with locality.

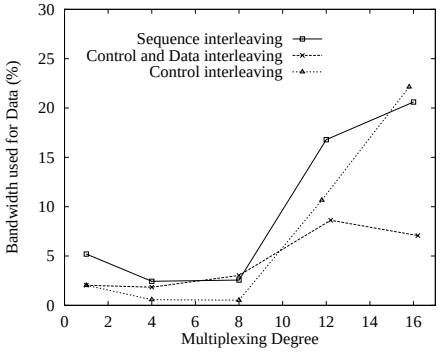


Figure 11. Short messages, RER with locality.

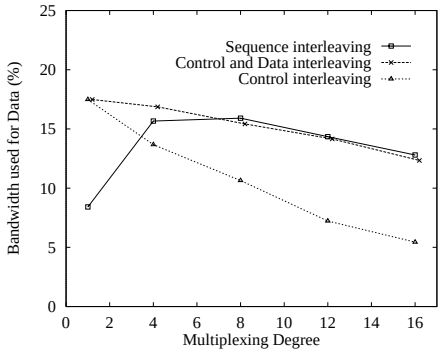


Figure 12. Long messages, RFE base cases.

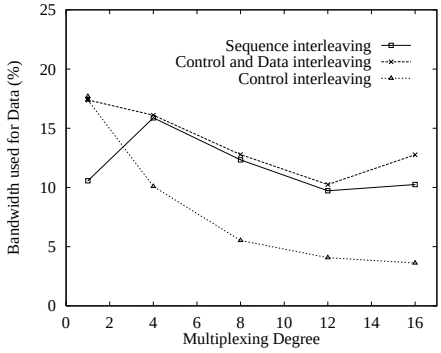


Figure 13. Long messages, RER base cases.

and a large degree of multiplexing. A timing analysis reveals that a circuit will be requested when control latency is less than the delay for circuit discovery. Performance is subsequently degraded by release processing.

With locality recognition, the results from individual simulation runs were highly variable when multiplexing at or above the optimal degree. At the minimum, performance slightly exceeded the base protocols. At best, performance approached the maximum attainable network utilization. (This maximum can be estimated from the number of packets to be sent and the number of time slots provided for data.)

From studies of the shuffle workload, we found two sources for this variation in performance. First, there can be multiple ways for circuit expiration and control cycle processing to synchronize into a repetitive pattern. Different patterns produce different values of steady state performance. Second, the random nature of the contention resolution algorithm causes the time to reach this synchronization to vary.

Figures 12 and 13 show the throughput of the base cases with long messages. For both protocols, the best performance is obtained with a multiplexing degree of one. Sequence interleaving performs poorly at this multiplexing degree because it provides less than half the data bandwidth of the other techniques. As the multiplexing degree increases, performance decreases. This is because the latency for changing a network state increases, resulting in unused time slots after the last packet of every message has been transmitted.

Figures 14 and 15 show the effect of locality recognition. RFE performance is relatively insensitive to the multiplexing degree. As with short messages, RER is able to identify the working set with a multiplexing degree of 12.

5 Conclusions

We have described two general classes of dynamic control protocols for circuit-switched networks, based on the use of a control cycle. The RFE protocol provides circuits for a fixed amount of time by automatically releasing network resources. The RER protocol allows circuits to remain as long as needed, at the cost of additional implementation complexity. RER is suited to applications that require guaranteed bandwidth. Both protocols can exploit communication locality of reference to increase their performance.

We investigated protocol performance for a looping parallel application in a banyan network. We found that multiplexing improves performance when the protocol has suffi-

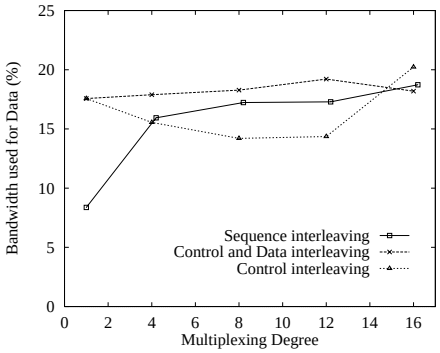


Figure 14. Long messages, RFE with locality.

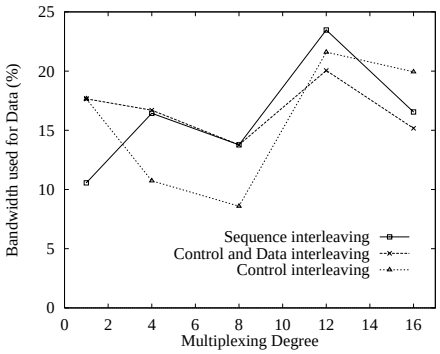


Figure 15. Long messages, RER with locality.

cient power to identify the communication working set and the multiplexing degree is sufficient to contain it. RER is more powerful than RFE since it reduces contention between requests for new circuits. In some cases, this may allow the control bandwidth allocation to be reduced.

Without locality recognition, best performance is often achieved using control operations to change the circuits provided by a single network state. Locality recognition provides a mechanism for identifying the communication working set of a parallel program and placing it into a set of states. When the number of states is sufficient, the capabilities of multiplexing hardare can be exploited to improve communication performance.

References

[1] S. Araki et al. Experimental free-space optical network for massively parallel computers. *Applied Optics*, 35(8):1269–81, March 1996.

[2] R. D. Chamberlain, M. A. Franklin, R. R. Krchnavek, and B. H. Baysal. Design of an optically-interconnected multiprocessor. In *Proceedings of MPPOI '98*, pages 114–122. IEEE Comp. Soc. Press, Los Alamitos, California, June 1998.

[3] D. M. Chiarulli, S. P. Levitan, R. G. Melhem, J. P. Teza, and G. Gravenstreter. Partitioned optical passive star (POPS) multiprocessor interconnection networks with distributed control. *Journal of Lightwave Technology*, 14(7):1601–1612, July 1996.

[4] 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.

[5] 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.

[6] P. Dowd, K. Bogineni, K. Aly, and J. Perreault. Hierarchical scalable photonic architectures for high-performance processor interconnection. *IEEE Transactions on Computers*, 42(9):1105–1120, September 1993.

[7] G. Liu, K. Lee, and H. Jordan. TDM hypercube and TWDM mesh optical interconnections. In *GLOBECOM '94: Proceedings of the IEEE Global Telecommunication Conference*, pages 1953–1957, Piscataway, NJ, 1994. IEEE.

[8] B. Mukherjee. WDM-based local lightwave networks part I: Single-hop systems. *IEEE Network*, 6(3):12–27, May 1992.

[9] 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.

[10] C. Salisbury. *Online Control of Multiplexed, Circuit-switched, Optical Interconnection Networks*. PhD thesis, University of Pittsburgh, December 1998.

[11] C. Salisbury and R. Melhem. Distributed, dynamic control of circuit-switched banyan networks. In *Proceedings of IPPS/SPDP '98*, pages 156–161. IEEE Comp. Soc. Press, Los Alamitos, California, March 1998.

[12] R. Thompson. The dilated slipped banyan switching network architecture for use in an all optical local area network. *IEEE Journal of Lightwave Technology*, 9(12):1780–1787, December 1991.