On the Performance of a Deadlock-free Routing Algorithm for Boolean n-Cube Interconnection Networks with Finite Buffers[1]

Ming-yun Horng and Leonard Kleinrock
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024-1596
(213) 825-3643

Abstract

This paper presents a mathematical model for evaluating the performance of a deadlock-free routing algorithm for Boolean n-cube interconnection networks with finite buffers. With this deadlock-free routing algorithm, all messages entering the network will be delivered correctly to their destinations without discards or deadlocks. We solve for the throughput of the network, the average message delay and the probability of acceptance of an input message. We determine the effect of the buffer size on performance and show that only a few buffers in each node are essential to yield good performance. We also show that the throughput of the network does not degrade even when the network is saturated.

Keywords: Boolean n-Cube Networks, Hypercubes, Interconnection Networks, Multiprocessor systems, Routing Algorithm, Performance Analysis.

1 Introduction

A major problem in designing a multiprocessor system is to construct a fast and efficient interconnection network among the processors. Many interconnection networks have been proposed [7], [18]. There is no single network that is considered the best for all applications. Recently, the Boolean n-cube interconnection network (also known by such names as binary n-cube, cosmic cube and hypercube) has been drawing much attention due to its powerful topological properties and its application to high speed switching networks. Several research and commercial systems have been built based on this type of network [17], [10], [8].

One important requirement is that the interconnection network should be deadlock-free. Deadlocks may occur as node buffers become full. A node with full input buffers also blocks its neighbors' output channels and increases the chance of the neighbors becoming blocked. When the movement of messages comes to a halt, the system crashes. Several deadlock prevention techniques based on the concepts of removing or avoiding cycles of channel dependency have been developed [6], [14].

However, with current VLSI technology, the channel speed in an interconnection network is approaching one gigabit per second [2]. At this high speed, the amount of time that the routing algorithm can afford to spend in making routing decisions is severely constrained. The routing algorithm must be very simple.

In this paper, by exploiting the homogeneity of the network and applying the concepts of timestamps [4] and backtracking [8], we present a simple deadlock-free routing algorithm for finite-buffered Boolean n-cube networks. With this algorithm, most messages are transmitted over their shortest paths from source to destination nodes. In a few situations, when a node does not have enough free buffers to accept all the incoming transit messages, the node sends some of its buffered messages on less direct paths to avoid congestion. Moreover, every message carries a globally unique timestamp. The oldest message will make nonstop progress to its destination without being backtracked. As soon as the oldest message has been delivered, another message becomes the oldest. Thus, every message accepted into the network is guaranteed to arrive at its destination without loss.

Boolean n-cube networks have been analyzed in the literature [5], [16], [3], [1]. However, mathematical models that consider queueing effects are rare. Moreover, in a large scale interconnection network, the storage capacity of each node is limited. Thus, a model which assumes there are an infinite number of buffers in each node cannot reflect the actual behavior of the system. Abraham and Padmanabhan [1] developed a model considering finite buffers, however, in which messages can be lost during transmission if the buffers at an output channel are full.

This paper presents a mathematical model for evaluating the performance of an algorithm with backtracking for lossless finite-buffered Boolean n-cube interconnection networks. We solve for the throughput of the network, the average message delay and the probability of acceptance of an input message. We determine the effect of the buffer size on the performance. We show that only a few buffers in each node are essential to yield good performance. We also show that the throughput of the network does not degrade even when the network is saturated. The match between the model and the simulation results is extremely good.

[1]This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

2 Preliminaries

2.1 Operation of the Network

Topological properties of the Boolean n-cube network are discussed in [15], [17]. A Boolean n-cube network consists of 2^n nodes, each addressed by an n-bit binary number from 0 to $2^n - 1$. Nodes are interconnected in such a way that there is a link between two nodes if and only if their addresses differ in exactly one bit position. Every node has exactly n neighbors. A Boolean 4-cube network is shown in Figure 1.

Each node handles messages for several local processors. Communication among nodes is achieved by multi-hop message passing. The header of a message can be computed as the exclusive-OR of the message's source and destination addresses. This information indicates the dimensions the message must traverse before reaching its destination. A one-bit in the header corresponds to a valid channel for transmission. Whenever a message is sent along a valid channel, the corresponding one-bit is changed to zero and the message is one hop closer to its destination. When the header contains only zeroes, the message has arrived at its destination.

Also, a message can be sent along a non-valid channel while the corresponding bit is changed from zero to one. As a result, the message is sent one hop farther away from its destination. The message then needs an extra hop to move itself back along this dimension later before reaching the destination. We say a message is forwarded if it is sent along a valid channel. A message is said to be backtracked if it is sent along a non-valid channel.

Since the propagation delay between two neighboring nodes is very small, it is possible to operate the system synchronously. We assume time is divided into cycles with a duration which corresponds to the transmission time of a message to its neighbor. We assume that all messages are of the same fixed size. We further assume that a node is capable of sending messages along its n channels simultaneously. Thus, a node can receive up to n messages from its neighbors in a cycle. The communication error rate in a well protected interconnection network is assumed to be extremely small and negligible.

2.2 Deadlock-free Routing Algorithm

At the beginning of a cycle, each node of the network makes its two-phase routing decision as follows. In the first phase, every node randomly selects one of the one-bits from the header of every message the node currently has. If more than one message is selected with the same bit position, the one having the highest priority (to be defined later) is successfully assigned to the channel. Obviously, the random assignment of messages to channels in the first phase is not an optimal approach in terms of the number of messages transmitted in a cycle. However, in [9] we have shown that the mean delay of the random approach is fairly close to the lower bound. Moreover, the routing decision of this random approach can be made simultaneously for all messages in a node by a parallel circuit and is therefore a simple and rapid calculation.

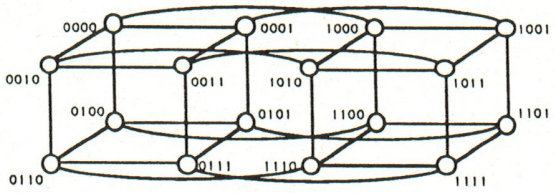

Figure 1: A Boolean 4-cube network

Let M be the buffer size of each node, where $M \geq n$. Suppose a node currently has i messages and j of them are successfully assigned to channels in the first phase. Then, $M - i + j$ buffers will be freed for any messages coming in from neighbors if there is no second phase. However, we wish to make available at least n buffers in case each of the node's n neighbors choose to send it a message in the cycle. So, in order to provide n free buffers when $M - i + j < n$, the node forces out $n - M + i - j$ messages with lower priority in the second phase. Since j channels have been assigned in the first phase, messages chosen in the second phase must be assigned only to the other $n - j$ channels. Note that $n - M + i - j \leq n - j$ in any case. If any message is assigned to a channel in the second phase with the corresponding bit in its header being one, then this message will be forwarded in the direction toward the destination. Otherwise, the message will be backtracked. A similar technique called "referral" is used in the Connection Machine [8]. Messages which fail in assignment in this cycle are kept in buffers for assignment in the next cycle. After making this two-phase routing decision, the node is ready to send messages to its neighbors.

Whenever two neighboring nodes are ready, they begin sending each other a message (if there is one) along the channel they share. We assume the channel is bidirectional. We further assume that, among these received messages, the ones destined for the node are delivered to local processors immediately. Other messages, known as *transit messages*, are saved in buffers for further transmission. After a node finishes exchanging messages with all of its neighbors, the node admits some input messages from its local processors and then continues to the next cycle.

The message priority is defined as follows. Each message is globally timestamped. The timestamp contains the time when the message was created and the source node address. Messages are queued and selected for routing in an order based on their timestamps, where older messages have higher priority over younger ones. If two messages are created at the same time, then the message with lower source address has higher priority over that with a higher source address. Since at least one of the stored messages is successfully assigned in the first phase, the message with highest priority always makes progress in every cycle. Whenever the oldest message has been delivered, another message becomes the oldest and proceeds without blocking. The network is deadlock-free.

3 Performance Analysis

In this section we develop an approximation to determine the throughput of the network, the average message delay, and the probability of acceptance of an input message.

3.1 Assumptions of the Model

We assume that a message's destination is uniformly distributed over the network, and that a local processor does not input messages to the node if the messages are for some other processors of the same node. Thus, we have

$$d_i = \text{Prob}[\textit{An input message has i 1-bits in its header}] = \frac{\binom{n}{i}}{2^n - 1}, \quad \text{for i} = 1, 2, \ldots, n$$

The expected number of hops a message must travel in the network is easily calculated as $\overline{d} = n\,2^{n-1}/(2^n - 1)$, which approaches $n/2$ when n is large. However, since messages can be backtracked during transmission, the number of hops actually traversed by a message can be larger than what is needed.

The arrivals of input messages from local processors to each node are assumed to be based on a geometric distribution with a generation rate of λ messages per cycle. We let g_i be the probability that a node's local processors generate i messages in a cycle. That is,

$$g_i = (1-\alpha)\,\alpha^i, \text{ where } 0 < \alpha = \frac{\lambda}{1+\lambda} < 1.$$

We note that in the case of destination nodes being uniformly distributed over the network, the throughput of a node must be less than 2 messages per cycle. However, some of these messages might be rejected by the node if the node does not have enough buffers. We define P_A as the probability of acceptance of an input message. Clearly, the communication load of the network is determined by λ and the set of probabilities d_i.

3.2 Imbedded Markov Chain Analysis

The Boolean n-cube network is assumed to be decomposed into a set of statistically identical nodes. (clearly, one should not use this assumption in the case of unbalanced traffic.) Considering an arbitrary node separately, we have a bulk-arrival and bulk-service system as shown in Figure 2. Let M be the buffer size of the node. We assume there exists an equilibrium state for the node.

Let the sequence of random variables $X_0, X_1, X_2, \ldots$ form an imbedded Markov chain, where X_m is the number of messages the node has at the beginning of cycle m. An $(M+1) \times (M+1)$ matrix P, which represents the one-cycle transition probability matrix of the node, is defined as

$$P = [p_{i,j}],$$

where

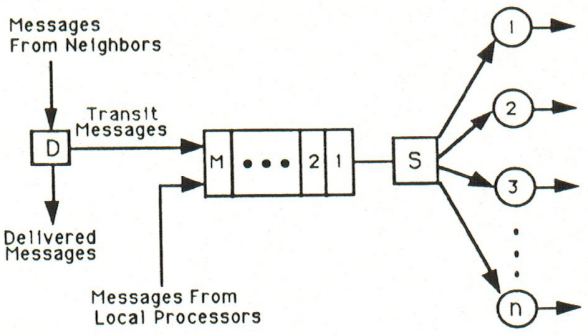

Figure 2: Structure of a Node

$$p_{i,j} = \lim_{m\to\infty} \text{Prob}\,[X_m = j \,|\, X_{m-1} = i]$$

We further define the steady state probability vector Π as

$$\Pi = [\pi_0, \pi_1, \pi_2, \ldots, \pi_M],$$

where

$$\pi_i = \lim_{m\to\infty} \text{Prob}\,[X_m = i] \text{ for i} = 0,1,2,\ldots,M$$

If the Markov chain is irreducible, aperiodic and recurrent nonnull, then Π can be uniquely determined through the following set of linear equations [11]:

$$\begin{aligned} \Pi &= \Pi P, \\ \sum_{i=0}^{M} \pi_i &= 1. \end{aligned} \tag{1}$$

3.2.1 Calculating the Matrix P

The key problem of this model is to calculate the transition probabilities $p_{i,j}$. Based on the description of the routing algorithm, it follows that the node repeatedly makes a routing decision, exchanges messages with its neighbors, and then admits some input messages from its local processors. However, without loss of generality, in the model we assume in each cycle all of the transit message are received by the node after all of its selected messages have been transmitted. See Figure 3.

We let $Y_0, Y_1, Y_2, \ldots$ be a sequence of random variables, where Y_m is the number of messages the node has in cycle m after the node has transmitted all of its currently selected messages, but before it has received any transit messages. We also let $Z_0, Z_1, Z_2, \ldots$ be a sequence of random variables, where Z_m is the number of messages the node has in cycle m after the node has received all of its transit messages, but before it has accepted any input messages.

Let us first determine the number of messages transmitted by the node in a cycle. Since one-bits in the header are assumed to be uniformly distributed and channel selection is made randomly, it follows that every channel of the node has an equal probability of being assigned in the first phase. We let $f_{i,j}$ be the probability that j messages are successfully assigned to channels in the first phase,

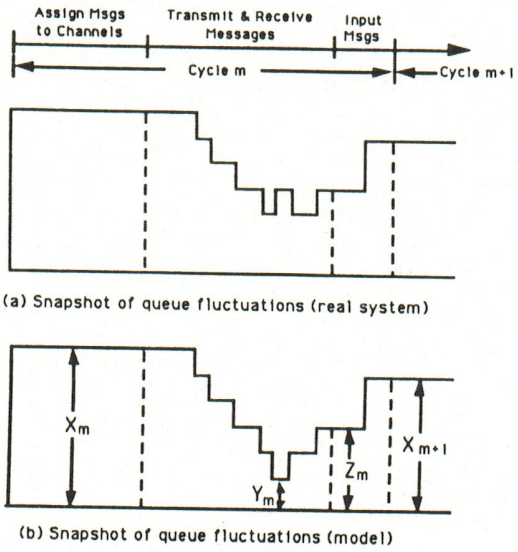

Figure 3: Snapshot of queue fluctuations in cycle m

given that the node currently has i messages. It can be shown [9] that

$$f_{i,j} = \begin{cases} \binom{n}{j} \sum_{k=0}^{j-1} (-1)^k \binom{j}{k} \left(\frac{j-k}{n}\right)^i & \text{if } \begin{cases} \text{if } i \geq 1 \text{ and} \\ 1 \leq j \leq \min(i,n) \end{cases} \\ 1 & \text{if } i = j = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Given that j messages are successfully assigned in the first phase, the second phase must choose $n - M + i - j$ messages if $i - j > M - n$. If we further let $f'_{i,j}$ be the probability that j messages are successfully assigned to channels at the end of both phases, given that the node currently has i messages, we have

$$f'_{i,j} = \begin{cases} f_{i,j} & \text{if } i - j < M - n \\ \sum_{k=1}^{n-M+i} f_{i,k} & \text{if } i - j = M - n \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

We now proceed to determine the number of transit messages received in a cycle. Recall that a transit message is a message received from a neighbor which needs further transmission. Assuming that the traffic load is evenly distributed and that any incoming channel is independent of any other incoming channel, and letting P_t be the probability of receiving a transit message from a channel and t_i be the probability of receiving i transit messages in a cycle, we have

$$t_i = \binom{n}{i} P_t^i (1 - P_t)^{n-i}, \text{ for i = 0,1,2,...,n.} \tag{4}$$

We solve for P_t later. We further define the following three $(M+1) \times (M+1)$ matrices:

$$D = [d_{i,j}],$$

where

$$d_{i,j} = \lim_{m \to \infty} \text{Prob}[Y_m = j \mid X_m = i].$$

$$R = [r_{j,k}],$$

where

$$r_{j,k} = \lim_{m \to \infty} \text{Prob}[Z_m = k \mid Y_m = j].$$

$$A = [a_{k,l}],$$

where

$$a_{k,l} = \lim_{m \to \infty} \text{Prob}[X_{m+1} = l \mid Z_m = k].$$

We find the following equations:

$$d_{i,j} = \begin{cases} f'_{i,i-j} & \text{if } 0 \leq j \leq i \leq M \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and

$$r_{j,k} = \begin{cases} t_{k-j} & \text{if } 0 \leq j \leq k \leq M \text{ and } k - j \leq n \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and

$$a_{k,l} = \begin{cases} (1-\alpha)\,\alpha^{l-k} & \text{if } 0 \leq k \leq l \leq M-1 \\ \alpha^{M-k} & \text{if } 0 \leq k \leq M \text{ and } l = M \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Finally, we have the one-cycle transition probability matrix P:

$$P = DRA. \tag{8}$$

Since $p_{i,i} \neq 0$ for $i = 0, 1, 2, ..., M$, the Markov chain is irreducible. It is not difficult to show that the Markov chain is also aperiodic and recurrent nonnull. Thus, the probability distribution π_i for $i = 0, 1, 2, ..., M$ can be determined from the set of linear equations described in Eq.1. However, the remaining problem is to determine the value for P_t.

3.2.2 Determining the Value for P_t

The value for P_t must satisfy the condition that in equilibrium the network message input flow equals the message output flow. We let u be the channel utilization, which is equal to the probability that a channel transmits a message in a cycle. Since a message can be assigned to a channel either in the first phase or in the second phase, we let u_1 be the probability that a channel transmits a message which is assigned in the first phase, and u_2 be the probability that a channel transmits a message which is assigned in the second phase. Clearly, $u = u_1 + u_2$. It is also clear that, given the probability distribution of $\pi'_i s$, we have

$$u = \sum_{i=1}^{M} \pi_i \sum_{j=1}^{\min(i,n)} f'_{i,j} \frac{j}{n}, \tag{9}$$

and

$$u_1 = \sum_{i=1}^{M} \pi_i \sum_{j=1}^{\min(i,n)} f_{i,j} \frac{j}{n}. \quad (10)$$

We further assume that every transmitted message has the same probability p of being assigned in the first phase and has the same probability q of being assigned in the second phase. That is,

$$p = \text{Prob}\left[\begin{array}{l}\textit{A message is assigned in the 1st phase,}\\ \textit{given that it is transmitted in the cycle.}\end{array}\right] = u_1/u$$

and

$$q = \text{Prob}\left[\begin{array}{l}\textit{A message is assigned in the 2nd phase,}\\ \textit{given that it is transmitted in the cycle.}\end{array}\right] = u_2/u,$$

where $p+q=1$. We note that if a message is assigned in the second phase, it must be assigned to a free channel which was not assigned in the first phase. Thus, if a message with i one-bits in its header is assigned to a channel in the second phase, it will be forwarded with probability $(i-1)/(n-1)$ and be backtracked with probability $(n-i)/(n-1)$. The state transition diagram for this case is given in Figure 4. The system is in state i if, at the beginning of a cycle, the message currently has i one-bits in its header. Let h_i be the expected number of steps to move from state i to state 0. We have the following recursive relations:

$$h_i = \begin{cases} 0 & \text{if } i = 0 \\ \left(p + \frac{i-1}{n-1}q\right) h_{i-1} + \left(\frac{n-i}{n-1}q\right) h_{i+1} + 1 & \text{if } i = 2,..,n-1 \\ h_{n-1} + 1 & \text{if } i = n. \end{cases}$$

It can be shown that

$$h_i = \sum_{j=n+1-i}^{n} \delta_j, \quad \text{for } i = 1,2,...,n \quad (11)$$

where

$$\delta_i = \begin{cases} 1 & \text{if } i = 1 \\ \prod_{j=1}^{i-1} \frac{\frac{j}{n-1}q}{1-\frac{j}{n-1}q} + \frac{n-1}{q} \sum_{m=1}^{i-1} \frac{1}{m} \prod_{j=m}^{i-1} \frac{\frac{j}{n-1}q}{1-\frac{j}{n-1}q} & \text{for } i = 2,...,n. \end{cases}$$

Let $\overline{h}$ be the expected number of hops actually traversed by a message. That is,

$$\overline{h} = \sum_{i=1}^{n} d_i h_i. \quad (12)$$

We further let P_e be the probability that a message from a neighbor is delivered to a local processor in the node. On average a message is expected to exit from the network after moving $\overline{h}$ hops. Thus, we have

$$P_e = 1/\overline{h}. \quad (13)$$

The message output rate of the network is given by

$$NnuP_e = Nnu/\overline{h}, \quad (14)$$

where $N = 2^n$ is the number of nodes in the network. Moreover, given the arrival process is geometric, it can be shown [9] that

$$P_A = 1 - \pi_M. \quad (15)$$

Thus, the message input rate to the network is given by

$$N\lambda P_A = N\lambda(1-\pi_M). \quad (16)$$

In equilibrium, the input rate equals the output rate. From Eqs. 16 and 14, we have that

$$\lambda(1-\pi_M) = nu/\overline{h}. \quad (17)$$

This equation must be satisfied by the given P_t and the calculated probability distribution π_i. In all the examples we have studied, we have found that the input rate is a decreasing function of P_t while the output rate is an increasing function; moreover, the input rate is larger than the output rate when P_t approaches 0 and that the input rate is smaller than the output rate when P_t approaches 1. Thus, in these examples, P_t exists and is unique. See Figure 5 and [9].

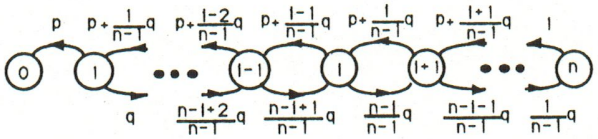

Figure 4: Transition diagram of finding i one-bits in the header

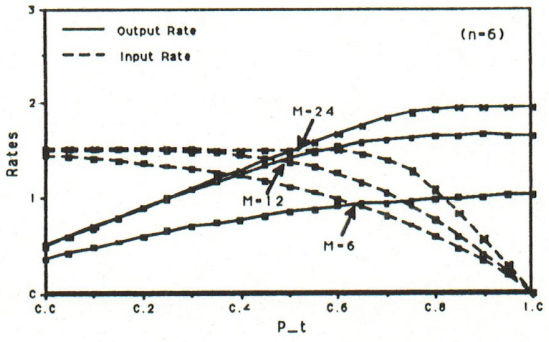

Figure 5: Input rate and output rate vs a given P_t

3.3 Delay and Throughput

The throughput of the network, γ, is defined as the total number of new messages the network accepts in a cycle. Thus,

$$\gamma = N\lambda(1-\pi_M) \quad (18)$$

The mean queue length in each node is given by

$$\overline{q} = \sum_{i}^{M} i\pi_i. \quad (19)$$

Applying Little's result [13], the average message delay is given by

$$T = \frac{N\bar{q}}{\gamma} \tag{20}$$

3.4 Effects of Backtracking

We now calculate the effects of backtracking. Let us define

P_f = Prob[a *channel forwards a message in a cycle*]
P_r = Prob[a *channel backtracks a message in a cycle*],

where $P_f + P_r = u$. The net progress made by a channel in a cycle is equal to $P_f - P_r$. We realize that whenever a message is backtracked, it is one hop farther away from its destination node. The message then needs an extra step of forwarding to compensate for this loss. We further let

C = *Number of channels in the network,*
K = *Number of cycles in a long time period, and*
S = *Throughput of the network in this period.*

For the whole network, the expected number of one-bits (in headers) which are changed to zeroes in K cycles is easily calculated as CKP_f. The expected number of zero-bits which are changed to ones is CKP_r. As a result, the expected number of one-bits "decreased" due to the network's transmission is $CK(P_f - P_r)$, which must be equal to the number of one-bits in the headers of input messages from local processors in this period when the network is in a steady state. Thus,

$$CK(P_f - P_r) = S\bar{d} \tag{21}$$

Moreover, the expected total number of hops made by all messages (forward or backward) in K cycles is

$$CK(P_f + P_r) = S\bar{h} \tag{22}$$

Thus, we immediately have the following equation:

$$\frac{\bar{h}}{\bar{d}} = \frac{P_f + P_r}{P_f - P_r}. \tag{23}$$

Solving for P_r and P_r, we have

$$P_r = \frac{u}{2}(1 - \frac{\bar{d}}{\bar{h}}), \tag{24}$$

$$P_f = \frac{u}{2}(1 + \frac{\bar{d}}{\bar{h}}). \tag{25}$$

The net progress of a channel in a cycle is then given by

$$P_f - P_r = u\frac{\bar{d}}{\bar{h}}. \tag{26}$$

4 Model Verification and Discussion

The accuracy of the mathematical model is verified by comparing it with simulations. In this section, we present several results of these simulations for a Boolean 6-cube network with various numbers of buffers in each node. Other results are reported in [9]. We note that the match between the simulated results and the model is extremely good.

Figure 6 shows the utilization, the probability of forwarding, and the net progress of a channel when $M = 10$. The net progress is also compared with that in the infinite buffered network where the net progress equals the channel utilization. We note that the effect of backtracking is not serious even if the buffer size is relatively small. Figure 7 presents the probability of acceptance of an input message for various buffer sizes. Again, we see that small buffers are enough to accept most of the input messages. Figure 8 shows the average message delay. We realize that when the generation rate is low, a node with a large buffer simply behaves as a node with an infinite buffer space. Thus, an increase in buffer size does not affect the delay. However, when the generation rate is high, more buffers admit more input messages. The queue grows as the buffer size increases. Thus, the average message delay also increases. When the buffer size in each node is very small (eg. 6 in this case), the average message delay is also larger because messages are likely to be backtracked. It is important to note in Figure 9 that even if the new message generation rate is much larger than that can be accepted by the network, the throughput of the network does not degrade.

5 Optimization Issues

In most queueing systems, two performance measures, response time and throughput, compete with each other. Typically, by raising the throughput of the system, which is desirable, the mean response time is also raised, which is not desirable. Moreover, we wish to consider the blocking of newly generated messages from local processors. We combine these three performance measures into a single measure, *power*, which is given as follows [12].

$$Power = \frac{Throughput}{Mean\ Response\ Time}(1 - B),$$

where B is the blocking probability. For our system we have

$$Power = \frac{\gamma}{T}P_A,$$

A system is said to be operating at an optimal point if the power is maximized. In Figure 10, we show the power as a function of λ for various buffer sizes; the peak of each curve identifies the optimal generation rate for each buffer size. This result is able to serve as a guide to network flow control. In Figure 11, we observe that an increase in buffer size increases the power. However, we note that assigning a large number of buffers to a node cannot significantly improve the performance. If we

wish to consider the cost of the buffer, we can further divide the power by the buffer size; using this modified measure, we show in Figure 12 that small buffers yield good performance.

6 Conclusion

We have presented a mathematical model for evaluating the performance of a deadlock-free routing algorithm for Boolean n-cube networks with finite buffers. This algorithm is simple enough to be implemented in hardware. We have shown that only a few buffers in each node are essential to yield good performance. We have also shown that the throughput of the network does not degrade even when the network is saturated.

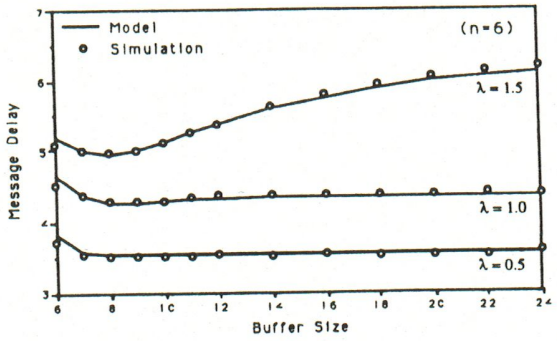

Figure 8: Average message delay vs buffer size

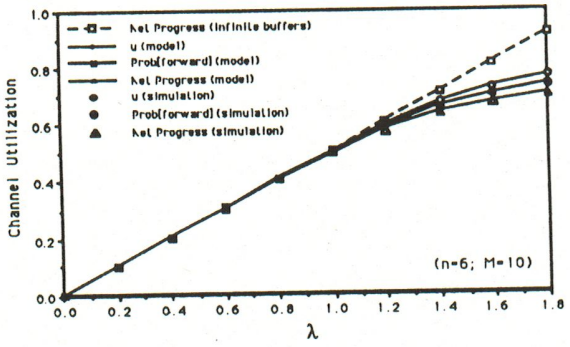

Figure 6: Channel utilization and prob. of forwarding

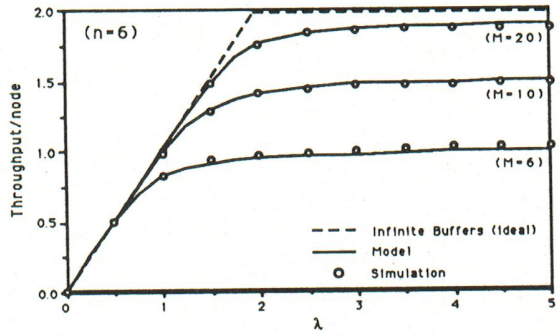

Figure 9: Throughput vs new message generation rate

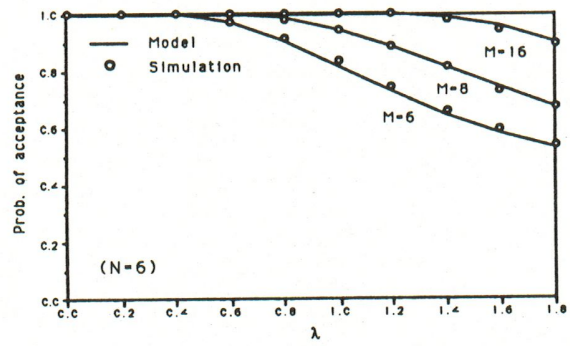

Figure 7: Prob. of acceptance of an input message

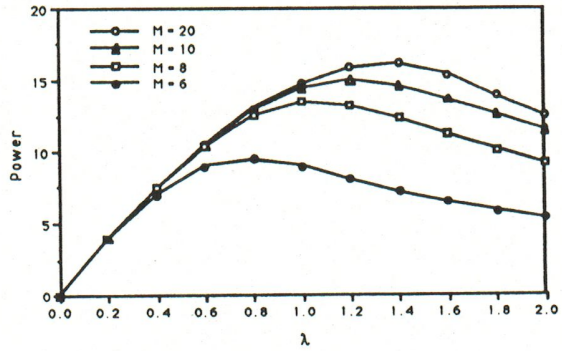

Figure 10: Power vs new message generation rate

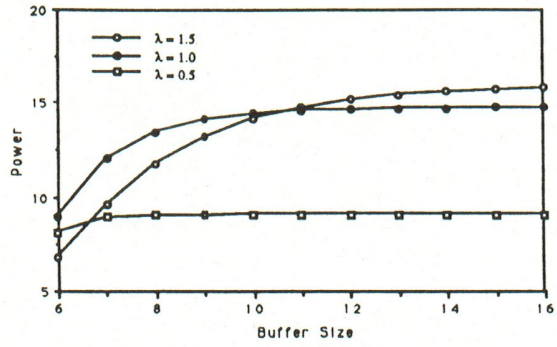

Figure 11: Power vs buffer size

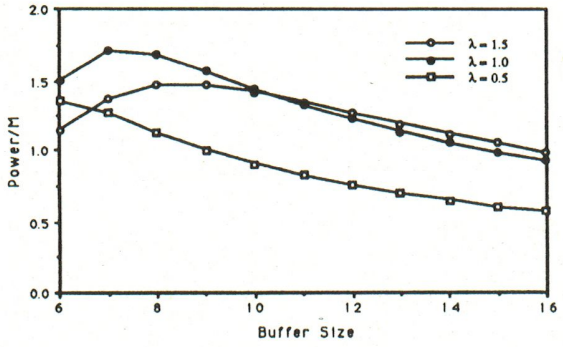

Figure 12: Power/M vs buffer size

References

[1] S. Abraham and K. Padmanabham, "Performance of Direct Binary n-Cube Networks for Multiprocessors," *IEEE Trans. Comput.*, vol. C-38, pp. 1000-1011, July 1989.

[2] W. C. Athas and C. L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *IEEE Comput.*, pp. 9-24, Aug. 1988.

[3] L. N. Bhuyan and D. P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Trans. Comput.*, vol. C-33, pp. 323-333, Apr. 1984.

[4] J. Balzewicz, J. Brzezinski and G. Gambosi, "Time-Stamp Approach to Store-and-Forward Deadlock Prevention," *IEEE Trans. Commun.*, vol. COM-35, pp. 490-495, May 1987.

[5] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks," *IEEE Trans. Comput.*, vol.C-39, pp. 775-785, June 1990.

[6] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Comput.*, vol. C-36, pp. 547-553, May 1987.

[7] T. Feng, "A Survey of Interconnection Networks," *IEEE Trans. Comput.*, vol. C-30, pp. 12-27, Dec. 1981.

[8] W. D. Hillis, *The Connection Machine*, MIT Press, 1985.

[9] M.-Y. Horng, *Performance Analysis of the Boolean n-Cube Interconnection Network for Multiprocessors*, Ph.D. Dissertation, Comput. Sci. Dep., Univ. California, Los Angeles, 1991.

[10] Intel Scientific Computers, *iPSC User's Guide*, No. 175455-001, Santa Clara, Aug. 1985.

[11] L. Kleinrock, *Queueing Systems, Vol. I: Theory*, John Wiley and Sons, New York, 1975.

[12] L. Kleinrock, "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," *Int. Conf. on Commun.*, pp. 43.1.1-43.1.10, June 1979.

[13] J. D .C. Little, "A Proof of the Queueing Formula $L = \lambda W$", *Oper. Res.*, vol. 9, pp. 383-387, May 1961.

[14] P. M. Merlin and P. J. Schweitzer, "Deadlock Avoidance : Store-and-Forward Deadlock," *IEEE Trans. Commun.*, vol. COM-28, pp. 345-354, Mar. 1980.

[15] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Comput.*, vol. C-37, pp. 867-872, July 1988.

[16] Y. Saad and M. H. Schultz, "Data Communication in Hypercubes," *J. Parallel Distrib. Comput.*, vol. 6, pp. 115-135, 1989.

[17] C. L. Seitz, "The Cosmic Cube,", *Commun. ACM*, vol. 28, pp. 22-33, Jan.1985.

[18] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing*, Lexington Books, 1985.