UNICAST-BASED BROADCAST: AN ANALYSIS FOR THE HYPERCUBE WITH ADAPTIVE ROUTING

A. Shahrabi, M. Ould-Khaoua, L. M. Mackenzie
Computing Science Department, Glasgow University, Glasgow, UK
Tel: +44 141 339 8855 ext. 0914,6056,4262
{alireza,mohamed,lewis}@dcs.gla.ac.uk

Keywords
Interconnection Networks, Wormhole Switching, Adaptive Routing, Performance Modelling, Broadcast Communication.

ABSTRACT

Many analytical models of wormhole-routed networks have been proposed over the past few years. Most of these models, however, have been developed for unicast (or point-to-point) communication. There has been comparatively little activity in the area of analytical models of collective communication, such as broadcast. As a result, most existing studies have relied on simulation to evaluate the performance merits of collective communication algorithms. This paper presents an analytical model for predicting broadcast latency in the hypercube. Results obtained through simulation experiments show that the model exhibits a good degree of accuracy in predicting message latency under different working conditions.

1. INTRODUCTION

Many algorithms have been proposed for broadcast communication in wormhole-routed networks over the past few years [11], [12], [14]. Among these, *unicast-based* broadcast algorithms have been widely considered due to their simplicity and ease of implementation. Since these rely on the routing algorithm employed for unicast communication to route broadcast messages, they do not require any changes to router hardware [10]. However, it is critical that when proposing a new algorithm for collective communication operation, we evaluate it with accurate modeling of the underling routing. Analytical modelling offers a cost-effective and versatile tool that can help designers to assess the performance merits of such algorithms to ensure successful introduction in future multicomputers.

Analytical models of wormhole-routed networks have been widely reported in the literature, e.g. [2], [3], [5] and [13]. However, all these models have been discussed in the context of unicast communication. Previous research studies on collective communication have focused primarily on the design of efficient algorithms in wormhole-routed networks [11], [14] and there has been little work on the development of analytical models for these. This paper presents an analytical model to compute broadcast message latency in wormhole-routed hypercubes. The broadcast algorithm considered in this study is based on the unicast-based approach described in [7], [11], [12] with both broadcast and unicast messages routed according to Duato's adaptive algorithm [6]. The rest of the paper is organised as follows. Section 2 reviews some preliminary background that will be useful for the subsequent sections. Section 3 describes the analytical model while Section 4 validates the model through simulation. Finally, Section 5 concludes the study.

2. PRELIMINARIES

Broadcast algorithms reported in the literature have been discussed in the context of two router structures, notably the *multiple-port* and *single-port* models [7], [12]. The former enables copies of the same broadcast message to be injected into the network through different output channels concurrently, while the latter injects them sequentially one at a time. This study focuses on the multiple-port model, but with a few simple modifications, it can be easily adapted to the single-port case.

Our present study focuses on a unicast-based broadcast algorithm that produces a *spanning binomial tree* [7] based on the concept of recursive doubling; a spanning tree is a connected graph that spans the nodes of the graph, forming a tree with no cycles. To broadcast a message, a node needs to transmit the message along a spanning tree rooted at its own location. Using this algorithm, the number of start-ups increases logarithmically with the number of nodes. Each node in the system will receive the broadcast message and generate new copies to send them to its own nearest neighbors. The algorithm guarantees that every node will receive the message exactly once and in no later than *n* communication steps.

Abraham and Padmanabhan [1] have shown that when the branches of the broadcast tree are constructed in the same order (e.g. in an increasing order of network dimensions) the number of messages that cross each channel varies severely, resulting in an unbalanced traffic on network channels. To overcome this problem they have suggested assigning a different dimension as a base for every new broadcast tree. The base dimension can be selected at random or in a round-robin fashion. As has been shown in [1], this improves the traffic balance in the network, and achieves higher throughput. The rest of this paper describes an analytical model for computing the broadcast latency in wormhole-routed multiport hypercubes, using the spanning binomial algorithm that incorporates Abraham and Padmanabhan's suggestion. Hereafter we will refer to this algorithm as the broadcast algorithm. Details of the router structure used in the analysis can be found in [15].

3. THE ANALYTICAL MODEL

The model is based on the following assumptions, which are commonly accepted in the literature [1], [2], [13].

a) There are two types of messages in the network: "broadcast" and "unicast". A broadcast message is delivered to every node in the network using the broadcast algorithm described in Section 2. A unicast message is sent to other nodes in the network with equal probability. When a message is generated in a given source node, it has a finite probability β of being a broadcast and probability $(1-\beta)$ of being a unicast message. When $\beta = 1$ a pure broadcast traffic is defined, while $\beta = 0$ specifies a purely uniform traffic pattern. A similar traffic model has also already been used in [1].

b) Nodes generate traffic independently of each other, following a Poisson process with a mean rate of λ_g messages/cycle. The mean generation rate of the broadcast messages is $\lambda_{s_b} = \beta\lambda_g$ and that of unicast is $\lambda_{s_u} = (1-\beta)\lambda_g$.

c) Message length is *M* flits, each of which is transmitted in one cycle across the physical channel.

d) A local queue in a given source node has infinite capacity. Moreover, messages are transferred to the local PE as soon as they arrive at their destinations.

e) *V* virtual channels are used per physical channel. According to Duato's adaptive routing algorithm [6], class *a* contains $(V-1)$ virtual channels, which are crossed adaptively, and class *b* contains one virtual channel, which is crossed deterministically (e.g. in an increasing order of dimensions). Let the virtual channels belonging to class *a* and *b* be called the adaptive and deterministic virtual channels respectively. When there is more than one available adaptive virtual channel, a message chooses one at random.

The broadcast latency refers to the elapsed time from when a source node sends out the first copy of its broadcast message to its neighbouring nodes until the last destination in the network receives a copy. Many existing studies [1], [9], [11] have used broadcast latency as a metric to assess the performance merits of different broadcast algorithms because of its great influence on overall application speedup. The rest of this section describes how the proposed analytical model computes the broadcast latency for the hypercube network. Although the analysis focuses on the broadcast latency, the latency for unicast messages can easily be computed since our model already determines all the necessary information concerning these messages, e.g., waiting times at the source node and network channels.

The broadcast algorithm guarantees that each node in the network receives a copy of the broadcast message in no longer than *n* broadcast steps, corresponding to the height of the broadcast tree. The broadcast latency is composed of *n* latencies, each of which accounts for the time to send a broadcast message one step down in the tree. Let us refer to the broadcast message that crosses from one level of the broadcast tree as "one-step broadcast message" and let $\overline{L}_b$ denote the corresponding mean latency. The mean broadcast latency can be written as

$$Latency = n\,(\overline{L}_b + \Delta) \tag{1}$$

where Δ denote the start-up latency. The mean latency of a one-step broadcast message, $\overline{L}_b$, is composed of the mean network latency, $\overline{S}_b$, i.e. the time to make one hop in the network, and the mean waiting time seen by a message in the source node before entering the network, $\overline{W}_s$. However, to model the effects of virtual channel multiplexing the mean one-step broadcast message latency has to be scaled by a factor, $\overline{V}$, representing the average degree of virtual channels multiplexing that takes place at a given physical channel. Therefore, we can write $\overline{L}_b$ as

$$\overline{L}_b = (\overline{S}_b + \overline{W}_s)\,\overline{V} \tag{2}$$

Before describing how to determine the quantities $\overline{S}_b$, $\overline{W}_s$, and $\overline{V}$, we determine first the traffic rate, λ_c, on a given network channel.

Calculation of λ_c **:** All network channels have equal traffic rates due to adaptive routing, which distributes traffic evenly across network channels, the uniform traffic pattern for unicast messages, and the balanced broadcast traffic resulting from the broadcast algorithm. According to the broadcast algorithm, a broadcast message is replicated at various stages in the spanning tree. A replicated message is put in the local queue of the node, to be injected later across the required output channel. So, a source node generates messages with three different rates: unicast messages with a rate of $\lambda_{s_u} = (1-\beta)\lambda_g$, broadcast messages with a rate of $\lambda_{s_b} = \beta\lambda_g$, and replicated messages with a rate of λ_{s_r} . Given that a source node has generated a broadcast message, the probability that a particular node in the network, other than the source node, will replicate the broadcast message and deliver a copy to at least one of its neighbouring nodes is $(2^{n-1}-1)/(2^n-1)$. Since there are (2^n-1) other nodes in the network and the generation rate of broadcast messages is $\lambda_{s_b} = \beta\lambda_g$, the rate of replicated messages originating from a given node is given by:

$$\lambda_{s_r} = (2^{n-1}-1)\lambda_{s_b} = (2^{n-1}-1)\beta\lambda_g \tag{3}$$

Consider now an output channel. The traffic rate, λ_c, on the channel consists of three different traffic rates given by

$$\lambda_c = \lambda_{c_u} + \lambda_{c_b} + \lambda_{c_r} \tag{4}$$

where λ_{c_u} , λ_{c_b} and λ_{c_r} are the traffic rates due to unicast, broadcast, and replicated messages, respectively. To compute λ_{c_u} , consider a generated unicast message that needs to cross *i* dimensions $(1 \le i \le n)$ to reach its destination. The number of nodes that the message can reach after making *i* hops is $\binom{n}{i}$. Therefore, the probability, p_i , that a unicast message crosses *i* dimensions to reach its destination is given by

$$p_i = \binom{n}{i} \Big/ 2^n - 1 \tag{5}$$

The average number of dimensions that a unicast message crosses to reach its destination can therefore be written as

$$\overline{d} = \sum_{i=1}^{n} i\,p_i = \frac{n}{2}\frac{N}{N-1} \tag{6}$$

Since a router in the hypercube has *n* output channels and a node generates, on average, $\lambda_{s_u} = (1-\beta)\lambda_g$ unicast messages in a cycle, then the traffic rate, λ_{c_u}, of unicast messages received by each channel in the network is simply

$$\lambda_{c_u} = (1-\beta)\lambda_g \bar{d} \big/ n \tag{7}$$

A given source node generates broadcast messages with a rate $\lambda_{s_b} = \beta\lambda_g$. Since a copy of the broadcast message has to be sent to the *n* neighbouring nodes through the *n* output channels, the rate of broadcast traffic on a given channel is given by

$$\lambda_{c_b} = \beta\lambda_g \tag{8}$$

In order to compute the traffic rate, λ_{c_r}, due to replicated broadcast messages we need to know the mean number of replications that a given node performs in a broadcast operation. After the source node sends its broadcast message to its *n* neighbouring nodes, each neighbour replicates the message and sends a copy to at least one of its adjacent nodes. The subsequent nodes replicate the message (*n*-2), (*n*-3), …, 0 times until the message reaches all the nodes. The number of replication varies from one node to another depending on the node position in the broadcast tree. So, the probability that a broadcast message is replicated *i* times $(0 \le i \le n-1)$ when it reaches an intermediate node is given by

$$P_{r_i} = 2^{n-i-1} \big/ 2^n - 1 \tag{9}$$

Hence, the mean number of replication of a broadcast message in a given node can be expressed as

$$\bar{\omega} = \sum_{i=0}^{n-1} i \ P_{r_i} = \sum_{i=0}^{n-1} i \ \frac{2^{n-i-1}}{2^n - 1} \tag{10}$$

Given that a replicated message can be sent over output channel with equal probability, the traffic rate of replicated messages on each channel is given by

$$\lambda_{c_r} = \frac{\bar{\omega}}{n}\lambda_{s_r} = \frac{\bar{\omega}}{n}(2^{n-1} - 1)\beta\lambda_g \tag{11}$$

Calculation of $\bar{S}_b$**:** The mean network latency of a one-step broadcast message, $\bar{S}_b$, consists of two parts: one is the delay due to the actual message transmission time, and the other is the time due to blocking in the network. Since a one-step broadcast message makes one hop to reach the next destination node, $\bar{S}_b$ can be written as

$$\bar{S}_b = M + B_b \tag{12}$$

where *M* is the message length and B_b is the mean blocking time seen by the message as it crosses an output channel. Since a one-step broadcast message reaches the next destination in a single hop, it can use only one specific output channel to reach its destination. As a result, the message suffers blocking when all the adaptive virtual channels and the deterministic virtual channel belonging to the output channel are busy. Since adaptive routing distributes traffic evenly across network channels, the message sees the same mean waiting time, $\bar{W}_c$, to acquire a virtual channel at an output physical channel, regardless of its position in the network. Let P_v denote the probability that *v* virtual channels at a physical channel are busy (P_v is calculated below). Given that a one-step broadcast message is blocked when all the *V* virtual channels at the required output channel are busy, the mean blocking time, B_b, can be written as

$$B_b = P_V \bar{W}_c \tag{13}$$

To determine the mean waiting time to acquire a virtual channel, $\bar{W}_c$, in the event of blocking, a physical channel is treated as an M/G/1 queue with a mean waiting time of [8]

$$\bar{W}_c = \rho \bar{S}(1 + C_{\bar{S}}^2) \big/ 2(1-\rho) \tag{14}$$

$$\rho = \lambda_c \bar{S} \tag{15}$$

$$C_{\bar{S}}^2 = \sigma_{\bar{S}}^2 \big/ \bar{S}^2 \tag{16}$$

where λ_c is the traffic rate on a network channel, $\bar{S}$ is the mean service time, and $\sigma_{\bar{S}}^2$ is the variance of the service time distribution. The traffic rate λ_c is given by equation 4, and we now compute the other two quantities, $\bar{S}$ and $\sigma_{\bar{S}}^2$. One-step broadcast and unicast messages see different network latencies time as they cross a different number of channels to reach their destinations. The former see the mean network latency, $\bar{S}_b$, given by equation 12, and the latter see the mean network latency for a unicast message, $\bar{S}_u$. The mean service time seen by an arbitrary message considering both broadcast and unicast possibilities with their appropriate weights, is given by

$$\bar{S} = \frac{\lambda_{c_b} + \lambda_{c_r}}{\lambda_c}\bar{S}_b + \frac{\lambda_{c_u}}{\lambda_c}\bar{S}_u \tag{17}$$

To simplify the development of our model while maintaining accuracy in predicting message latency we use a suggestion applied by Draper and Ghosh [5] for computing the variance of the service time. Since the minimum service time at a channel is equal to the message length, the variance of the service time distribution can be approximated as

$$\sigma_{\bar{S}}^2 = (\bar{S} - M)^2 \tag{18}$$

As a result, the mean waiting time becomes

$$\bar{W}_c = \frac{\lambda_c \bar{S}^2 \ (1 + \frac{(\bar{S} - M)^2}{\bar{S}^2})}{2(1 - \lambda_c \bar{S})} \tag{19}$$

Let S_{u_i} denote the network latency for an *i* hops unicast message $(0 \le i \le n-1)$. Since the probability of generating an *i*-hop message by a given source node is p_i (equation 5), averaging over all possible hops made by unicast message yields the mean network latency, $\bar{S}_u$, as:

$$\bar{S}_u = \sum_{i=1}^{n} p_i \ S_{u_i} \tag{20}$$

The network latency, S_{u_i}, for an *i*-hop unicast message is determined in a similar manner to that for a one-step broadcast message. Therefore we can write

$$S_{u_i} = (M + i) + \sum_{j=1}^{i} B_{u_j} \tag{21}$$

where B_{u_j} is the mean blocking time seen by a unicast message at the *j*-th hop channel $(1 \le j \le i)$ along its network path. A unicast message is blocked at the *j*-th hop channel when all the adaptive virtual channels at the remaining dimensions to be visited, and also the deterministic virtual channels at the lowest dimension still to be visited, according to deterministic routing, are busy [6]. As stated above in the case of broadcast messages, adaptive routing distributes network rate evenly on the network channels, and a message sees the same mean waiting time, $\overline{W_c}$ (given by equation 19) across all network channels, regardless of its position. However, it sees a different probability of blocking at each hop since the number of alternative paths from its current position to its destination changes from one hop to another. If P_{b_j} denotes the probability of blocking at the *j*th-hop channel the mean blocking time can be written as

$$B_{u_j} = P_{b_j}\overline{W_c} \tag{22}$$

To compute P_{b_j} we need to compute, firstly, the probability that all adaptive virtual channels in a dimension are busy, P_a, and, secondly, the probability that all adaptive and deterministic virtual channels in a dimension are busy, P_d. As has been shown in [2], P_a and P_d can be approximated as

$$P_a = \frac{P_{V-1}}{\begin{pmatrix} V \\ V-1 \end{pmatrix}} + P_V \tag{23}$$

$$P_d = P_V \tag{24}$$

where P_v is the probability that V virtual channels at a given physical channel are busy. When an *i*-hop message has reached its *j*-th hop channel, it can use on its next hop any one of the $(n-1)(i-j)$ adaptive virtual channels belonging to the physical channels in the remaining $(i-j)$ dimensions. It can also use any one of *V* virtual channels (one deteministic virtual channel and (*V*-1) adaptive virtual channels) at the lowest dimension still to be visited according to deterministic routing. Combining the above cases and using equations 23 and 24 yields the probability of blocking, P_{b_j}, as

$$P_{b_j} = P_a^{i-j} P_d \tag{25}$$

The above equations reveals several inter-dependencies between the different variables of the model. For instance, equation 17 shows that $\overline{S}$ is a function of $\overline{S}_u$ and $\overline{S}_b$, while equations 12 and 21 show that $\overline{S}_u$ and $\overline{S}_b$ are functions of $\overline{S}$. Since obtaining closed-form expressions for such interdependencies is generally difficult, the different variables of the model are computed using iterative techniques for solving equations [2].

Calculation of $\overline{W_s}$: The mean waiting time in the source node is calculated in a similar way to that for a network channel (equations 14-16). Therefore, using an M/G/1 queue for the injection channel in the source node gives the mean arrival rate and mean service time as follow

$$\lambda_s = \frac{\lambda_{S_u}}{n} + \lambda_{S_b} + \frac{\overline{\omega}}{n}\lambda_{S_r} \tag{26}$$

$$\overline{S}_s = \frac{\lambda_{s_b} + \lambda_{s_r}}{\lambda_{s_u} + \lambda_{s_b} + \lambda_{s_r}}\overline{S}_b + \frac{\lambda_{s_u}}{\lambda_{s_u} + \lambda_{s_b} + \lambda_{s_r}}\overline{S}_u \tag{27}$$

Approximating the variance of the service time distribution by $(\overline{S}_s - M)^2$ yields a mean waiting time at the source of

$$\overline{W}_s = \lambda_s \overline{S}_s^2 \left(1 + \frac{(\overline{S}_s - M)^2}{\overline{S}_s^2}\right) \Big/ 2(1 - \lambda_s \overline{S}_s) \tag{28}$$

Calculation of $\overline{V}$: The probability, P_v, that *v* adaptive virtual channels are busy in a physical channel can be determined using a Markovian model as shown in [4].

$$q_v = \begin{cases} 1 & v = 0 \\ q_{v-1}\lambda_c \overline{S} & 0 < v < V \\ q_{v-1}\dfrac{\lambda_c}{1/\overline{S} - \lambda_c} & v = V \end{cases} \tag{29}$$

$$P_v = \begin{cases} 1/\sum_{j=0}^{V} q_j & v = 0 \\ P_{v-1}\lambda_c \overline{S} & 0 < v < V \\ P_{v-1}\dfrac{\lambda_c}{1/\overline{S} - \lambda_c} & v = V \end{cases} \tag{30}$$

In virtual channel flow control, multiple virtual channels share the bandwidth of a physical channel in a time-multiplexed manner. The average degree of multiplexing of virtual channels, which takes place at a given physical channel, is given by [4]

$$\overline{V} = \sum_{i=1}^{V} i^2 P_i \Big/ \sum_{i=1}^{V} i\, P_i \tag{31}$$

4. SIMULATION EXPERIMENTS

The above model has been validated using a discrete-event simulator that performs a time-step simulation of network operations at the flit level. Each simulation experiment is run until the network reaches its steady state; that is until a further increase in simulated network cycles does not change the collected statistics appreciably. Statistics gathering was inhibited for the first 20000 messages to avoid distortions due to the startup transient. Extensive validation experiments have been performed for several combinations of network sizes, message lengths, different fractions of broadcast messages and number of virtual channels. The startup latency, Δ, varies from one practical machine to another, and has usually been considered as a constant value independent of network traffic. For the purpose of our present study this delay factor has been fixed at $\Delta = 1$ cycle. Obviously, such a figure has no effect on the validation process, and higher values can be easily incorporated in the model. For the sake of specific illustration, latency results are only presented for the following cases: network size $N = 2^6$, 2^7 and 2^8 nodes, number of virtual channels *V*=2 and 4, message length *M*=32, 64 and 128 flits and broadcast portion $\beta = 0.005$, 0.01 and 0.02.

Figures 1 depicts results for the mean broadcast latency predicted by the above analytical model plotted against those provided by the simulator as a function of the traffic injected by each node in the network. The horizontal axis in the figures represent the message generation rate of every node per cycle, λ_g, while the vertical axis shows the mean broadcast latency. The figures reveal that the simulation results closely match those predicted by the analytical model in the steady state regions (i.e. under light and moderate traffic) and even when the network starts to approach saturation. However, the discrepancies in the results near saturation are noticeable. This is due to the approximations

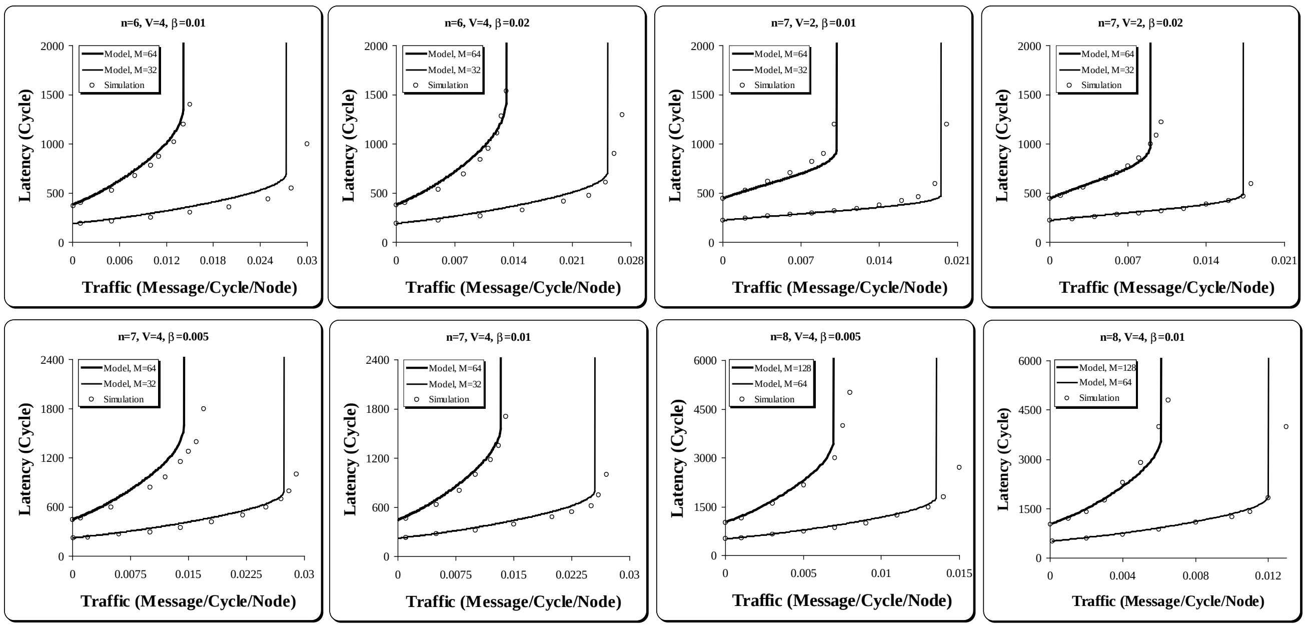

Figure 1: Validation of the model against simulation in 6, 7 and 8-dimensional hypercubes. Message length M = 32, 64 and 128, broadcast portion β = 0.005, 0.01 and 0.02, and number of virtual channels *V* = 2 and 4.

which have been made to simplify the development of the model, such as the one made in determining the variance of service time at a network channel. Nevertheless, we can conclude that the model produces accurate results in the steady state regions, and its simplicity makes it a practical evaluation tool that can be used to gain insight into the behaviour of wormhole-routed hypercubes in the presence of broadcast communication.

5. CONCLUSION

Although many broadcast algorithms have been proposed for wormhole-routed networks over the past decade, there has been little development of analytical models of these algorithms. This paper has presented just such a model capable of computing broadcast latency in wormhole-routed hypercubes under a number of reasonable assumptions. Simulation have shown close agreement with the results from the analytical model. An obvious continuation of this work would extend the model to other broadcast algorithms for the hypercube such as Double Tree (*DT*). Another line of progression would be to develop the analytical model for other network topologies (e.g. *k*-ary *n*-cubes and *n*-dimensional meshes) and for multidestination-based broadcast algorithms such as those based on the Base Routing Conformed Path (BRCP) methodology [14].

6. REFERENCES

[1] Abraham, S., Padmanabhan, K. Performance of the direct binary *n*-cube network for multiprocessors, *IEEE Trans. Computers*, vol. 38, no. 7, pp. 1,001-1,011, 1989.

[2] Boura, Y., Das, C.R., Jacob, T.M. A performance model for adaptive routing in hypercubes, *Proc. Int. Workshop Parallel processing*, pp. 11-16, 1994.

[3] Dally, W.J. Performance analysis of *k*-ary *n*-cubes interconnection networks, *IEEE TC* 39(6), pp. 775-785, 1990.

[4] Dally, W.J., Virtual channel flow control, *IEEE TPDS* 3 (2), pp. 194-205, 1992.

[5] Draper, J.T., Ghosh, J. A Comprehensive analytical model for wormhole routing in multicomputer systems, *J. Parallel & Distributed Computing* 32, pp. 202-214, 1994.

[6] Duato, J. A New theory of deadlock-free adaptive routing in wormhole routing networks, *IEEE TPDS* 4(12), pp. 320-1331, 1993.

[7] Duato, J., Yalamanchili, S., Ni, L. *Interconnection networks: An engineering approach*, IEEE Press, 1997.

[8] Kleinrock, L. Queueing Systems Vol. 1, John Wiley, New York, 1975.

[9] Lin, X., Mckinley, P., Ni, L. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. *IEEE TPDS* 5(8), pp. 793-804, 1994.

[10] Malumbres, M.P., Duato, J. and Torrellas, J. An efficient implementation of tree-based multicast routing for distributed shared memory multiprocessor, *ISPDP*, 1996.

[11] McKinley, P.K., Xu, H., Esfahanian, A.H. and Ni, L. M. , Unicast-based multicast communication in wormhole-routed networks, *IEEE TPDS* 5(12), pp.1,252-1,265, 1994.

[12] McKinley, P.K., Tsai, Y.J., Robinson, D.F. Collective communication in wormhole-routed massively parallel computers, *Computer*. 28(12), 39-50,1995.

[13] Ould-Khaoua, M. A performance model for Duato's fully-adaptive routing algorithm in *k*-ary *n*-cubes, *IEEE Trans. Computers*, vol. 42(12), pp. 1-8, 1999.

[14] Panda, D., Singal, S., Kesavan, R. Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths, *IEEE TPDS* 10(1), pp. 76-96, 1999.

[15] Shahrabi, A., et al. Analytical Modelling of Broadcast in Adaptive Wormhole-Routed Hypercube, Technical Report, Computing Science Department, University of Glasgow, 2000.