An Optimal $(d-1)$-Fault-Tolerant All-to-All Broadcasting Scheme for d-Dimensional Hypercubes

Siu-Cheung Chau *
Dept. of Physics and Computing, Wilfrid Laurier University,
Waterloo, Ontario, Canada, N2L 3C5
e-mail: schau@wlu.ca

Ada Wai-Chee Fu
Dept. of Computer Science and Engineering,
Chinese University of Hong Kong, Shatin, Hong Kong
e-mail: adafucse.cuhk.edu.hk

Abstract

All-to-all broadcasting (Gossiping) is the process of information dissemination in a communication network. Each member in the network has a message to transmit to all other members of the network. We proposed a k-fault-tolerant scheme for a faulty d-dimensional hypercube with $n = 2^d$ nodes where $0 \leq k < d$. The new scheme requires $n(n-1)$ fewer message transmissions and $(n-1)F\tau$ less time compared to previously proposed fault-tolerant all-to-all broadcasting schemes.

Keywords: All-to-all Broadcasting, Gossiping, Fault-tolerant, Hypercube.

1 Introduction

All-to-all broadcasting is the process of information dissemination in a communication network. Each node in the network has a message to transmit to all other nodes of the network. A k-fault-tolerant all-to-all broadcasting process is one which sends the messages out with enough redundancy so that the broadcasting can be completed even if k nodes has failed.

Two different models of communications, shouting and whispering[5] are considered. In the shouting model, a node can communicate simultaneously with all the adjacent nodes. That is, each node has all-port capability. In the whispering model, a node can only communicate with one adjacent node at any given time. Each node has one-port capability.

We assume that the communications are based on a message passing procedure where each communication channel is full-duplex and are sent in the store and forward mode. We only consider permanent node and link faults. In general, a node does not have any knowledge of the location of these faults. Furthermore, a node or a link is faulty if it cannot transmit any messages. It cannot corrupt messages.

In this paper, we will concentrate on fault-tolerant all-to-all broadcasting in multi-computer networks connected as hypercubes. A d-dimensional hypercube is a network with $n = 2^d$ nodes. The hypercube network is already commercially available and is being used for a variety of applications. Many researchers have investigated the all-to-all broadcasting (gossiping) problem. For a survey of gossiping papers, please refer to Hedetniemi and Liestman's paper[7].

For hypercubes, Bagchi et al.[1] conjectured that $2d$ steps are required to complete all-to-all broadcasting in a d-dimensional hypercube. Krumme[9] proved that their conjecture is incorrect by proposing a fast all-to-all broadcasting algorithm for a d-dimensional hypercube that requires only $1.88d$ steps. Since then, a lot of papers like Scott's[13], Petrini's[10], and Johnsson's[8] were published about all-to-all broadcasting in hypercubes. However, very few of them deal with all-to-all broadcasting when some nodes or links may have already failed.

Fraigniaud[5] proposed a asymptotically optimal k-fault-tolerant all-to-all broadcasting scheme for d-dimensional hypercubes where $0 \leq k < d$. His scheme is based on Johnsson and Ho's[8] arc-disjoint spanning tree. Let the time to send out a message be $T = \beta + F\tau$[8]. β is the start up time. τ is the time to send out one bit. F is the length of the message. If all the nodes start his algorithm simultaneously, his algorithm requires

*This research was supported by a research grant from the National Sciences and Engineering Research Council of Canada

$2d\beta + (k+1)(n-1)F\tau$ time for the whispering model and $(d+1)\beta + (n-1)F\tau$ for the shouting model. He also showed that for the whispering model, at least $(k+1)(n-1)F\tau$ propagation time and $(d+k+1)\beta$ starts up time is required for a k-fault-tolerant all-to-all broadcasting in a d-dimensional hypercube. For the shouting model, at least $(k+1)(n-1)F\tau/d$ propagation time and $(d+1)\beta$ start up time if $k = d-1$ or $d\beta$ start up time if $k < d-1$ are required.

In this paper, we proposed a new all-to-all broadcasting scheme for d-dimensional hypercubes that can tolerate up to $(d-1)$ node faults for both the whispering and shouting models. For the whispering model, the new scheme is faster and requires fewer message transmissions than Fraigniaud's scheme. The rest of the paper is organized as follows. In Section 2, we give the algorithm for a $(d-1)$-fault-tolerant one-to-all broadcasting in d-dimensional hypercubes. In Section 3, we present our k-fault-tolerant all-to-all broadcasting scheme. In Section 4, we compare the new scheme with previously proposed scheme. Section 5 contains the summary.

2 An optimal $(d-1)$-fault-tolerant one-to-all broadcasting scheme

A d-dimensional hypercube is a network with $n = 2^d$ nodes. Each node can be coded by a binary sequence of length d. Two nodes are connected if the binary sequences differ in exactly one position. Each node $v = x_1x_2...x_d$ is connected to d nodes. We call the link that connects $v = x_1x_2...x_d$ to the node $u = y_1y_2...y_d$, the link of dimension i if $x_i \neq y_i$ and $x_j = y_j$ for $j \in 1..d$ and $j \neq i$.

First, we assume that communication is carried out in the whispering model. The algorithm has two phases, the broadcast phase and the extended broadcast phase. The broadcast phase consists of d time units. In the first time unit of the broadcast phase, the originator o sends the broadcast message to an adjacent node through o_1, the link of dimension 1. In the 2^{nd} time unit, the originator and the node that got the message in time unit 1, send the message through the links of dimension 2. In the i^{th} time unit, the originator and all the nodes that already have the message send the message through the links of dimension i. The first phase requires d time units. After the first phase, every node will receive the message if the hypercube is non-faulty.

The second phase also consists of d time units. In the first time unit, every node sends a message through the links of dimension 1. In the second time unit, every node sends a message through the links of dimension 2. In the i^{th} time unit, every node sends a message through the links of dimension i. After the second phase, every node will have received d copies of the message if the hypercube is non-faulty.

It is obvious that every node in a non-faulty hypercube will receive the message after phase one of the new scheme. We will show that every node gets d node disjoint calling paths after both phases of the broadcast.

Consider a 4-dimensional hypercube with 16 nodes. Without loss of generality, let us assume that node 0(0000) is the originator. In time unit 1 of phase one, 0 sends to 8(1000). In time unit 2, 0 sends to 4, and 8 sends to 12. In time unit 3, 0 sends to 2, 4 sends to 6, 8 sends to 10, and 12 sends to 14. In the last time unit of phase one, 0 sends to 1, 2 sends to 3, 4 sends to 5, 6 sends to 7, 8 sends to 9, 10 sends to 11, 12 sends to 13, and 14 sends to 15.

For node 14, it gets a calling path from 0 to 8, 8 to 12, and 12 to 14 in the first phase. In the first time unit of phase two, node 14 gets another calling path from 0 to 4, 4 to 6, and 6 to 14. The calls from 0 to 4 and 4 to 6 are made in the first phase. The call from 6 to 14 is made in the first time unit of phase two. In the second time unit, another calling path is obtained from 0 to 2 in the first phase, 2 to 10 in the first call in the second phase, and from 10 to 14 in the second call of the second phase. The final calling path of node 14 is from 0 to 1 in the first phase, 1 to 9 in the first call of the second phase, 9 to 13 in the second call of the second phase, 13 to 15 in the third call of the second phase, and 15 to 14 in the fourth call of the second phase. The calling paths for node 14 are:

1. 0(0000) → 8(1000) → 12(1100) → 14(1110).
2. 0(0000) → 4(0100) → 6(0110) → $14(1110)^1$.
3. 0(0000) → 2(0010) → $10(1010)^1$ → $14(1110)^2$.
4. 0(0000) → 1(0001) → $9(1001)^1$ → $13(1101)^2$ → $15(1111)^3$ → $14(1110)^4$.

The calls with an ^{i} indicates they are calls made in the in the i^{th} time units of the second phase. The four calling paths are node disjoint.

Theorem 1 *Every node in the hypercube will have d node disjoint calling paths from the originator after the two phases of the broadcast.*

Proof: Without loss of generality, let $u = 00..0$ be the originator. The d-dimensional hypercube is made up of two $(d-1)$-dimensional hypercubes. The first $(d-1)$-dimensional hypercube consists of all the nodes with a zero in their leftmost bit. The second $(d-1)$-dimensional hypercube consists all the nodes with a one in their leftmost bit. In time unit 1 of phase one, the originator that resides in one of the $(d-1)$-dimensional hypercubes calls a node in the other $(d-1)$-dimensional hypercube. Subsequent calls of phase one are made within the two $(d-1)$-dimensional hypercubes. Thus, each node in the second hypercube has one calling path originated from 00..0 to 10..0 and the calling path also consists of calls within its own sub-cube. We call

this group of nodes M_1. Similarly, the $(d-1)$-dimensional hypercube that the originator is in, is also made up of two $(d-2)$-dimensional hypercubes. In time unit 2 of phase one, the originator calls a node in the other $(d-2)$-dimensional hypercube. Each node in the other $(d-2)$-dimensional hypercube has one calling path originated from 00..0 to 010..0 and the calling path also consists of calls within its own subcube. We call this group of nodes M_2. Hence, after d time units, the nodes are divided into M_i groups where $i = 1..d$ and the originator o. The group M_i is a $(d-i)$-dimensional hypercube.

Consider a node v in M_i. In phase two, the 1^{st} calling path of v is from the originator to the node 1000..00 in M_1, followed by broadcasting within M_1, and a call between v and a node in M_1 through the link in the 1^{st} dimension, v_1 in time unit 1 of phase two. The 2^{nd} calling path of v is from the originator to the node 0100..00 in M_2, followed by broadcasting within M_2, and a call between v and a node in M_2 through the link in the 2^{nd} dimension, v_2 in time unit 2. Similarly, The $(i-1)^{th}$ calling path of v is from the originator to the node with a 1 in the $(i-1)^{th}$ position 00..010..00 in M_{i-1}, followed by broadcasting within M_{i-1}, and a call between v and a node in M_{i-1} through the link in the $(i-1)^{th}$ dimension, v_{i-1} in time unit $(i-1)$ of phase 2. None of the links between v_1 and v_{i-1} are used in phase one of the broadcast. All the $(i-1)$ calling paths for v are disjoint.

If $i = d$, the node v will have $(d-1)$ edge and node disjoint calling path after $(d-1)$ time units. The d^{th} calling path of v is obtained from a call from the originator to v in phase one. Hence, v gets d disjoint calling paths after both phases if $i = d$.

If $i \neq d$, the i^{th} calling path of v is from the originator to a node in M_j, followed by broadcasting within M_j, where $i < j \leq d$, and a call between v and a node in M_j in time unit i. All the nodes in M_i get one more edge disjoint calling path in time unit i except node m where m is the node that received the call from the originator in time unit i of phase one. Node m does not get another node disjoint calling path for it is called by the originator again in time unit i.

Similar to the original d-dimensional hypercube, M_i can be divided into $d-i$ groups, $M_{i,i+1}$ to $M_{i,d}$ and the node m. The group $M_{i,j}$ is a $(d-j)$-dimensional hypercube. The i^{th} calling path of nodes in $M_{i,j}$ is from the originator to a node in M_j, followed by broadcasting within M_j, and calls between M_j and $M_{i,j}$.

For the node m, it will get an additional $d-i$ edge disjoint calling paths from $M_{i,j}$ to m in time unit j where $i+1 \leq j \leq d$, through the link in dimension j, m_j. Although the link m_j has already been used in phase one of the broadcast, it is used to send messages from m to nodes in $M_{i,j}$. In phase two, we are using it in the other direction. Hence, node m will have d node disjoint paths from the originator after both phases.

For the node v, let v be in $M_{i,k}$. As described above, v will get its $(i+1)^{th}$ to $i+k-1$ node disjoint calling paths from calls between v and a node in $M_{i,j}$ where $i+1 \leq j < k$ in time units $(i+1)^{th}$ to $i+k-1$.

In time unit $i+k$, again $M_{i,k}$ can be divided into $d-i-k$ groups, $M_{i,k,k+1}$ to $M_{i,k,d}$ and a single node m'. This process is repeated until v becomes a node in $M_{i,k,..,d}$ or v becomes the single node after a division. In both cases, the node v will have $(d-1)$ node disjoint calling paths after d time units of phase two. Adding the calling path of v obtained in the first phase, the node v has d node disjoint calling paths after both phases. $\triangle$

In phase one, each node may receive the message at most once. Hence, at most $n-1$ messages are sent. In each time unit in phase two, every node has to send a message to an adjacent node. Hence, n messages are sent. The total number of messages is $(n-1)+nd$. However, if we are a bit more careful in the extended broadcast, we can reduce the number of messages sent. This can be accomplished by not sending from a node v to a node u if node v has already sent a message to node u in phase one. $n-1$ messages are used in phase one. The total number can then be reduced by $n-1$. Similarly, an additional $n-1$ messages can be reduced by not sending in phase two through the links where nodes got their messages in phase one. The total number of messages required for a faulty hypercube is at most $((n-1)+nd) - 2(n-1) = (nd-n+1)$. If some nodes or links have already failed in the hypercube, the number of messages sent is even less.

If the hypercube is non-faulty, the messages sent through links of dimension d in phase two are not necessary. These messages are sent from $n/2$ nodes that received the message and from $n/2$ nodes that sent out the message in time unit d of phase one through links of dimension d. Hence, these nodes knew that the nodes that are connected to them through links of dimension d already got the message. It is not necessary for these nodes to send the message out in time unit d of phase two. Hence, only $2d-1$ time unit is required if the hypercube is non-faulty.

Fraigniaud[5] proved that for the whispering model, at least $(d+k+1)$ time units are required to achieve k-fault-tolerant broadcast in a d-dimensional hypercube for $1 \leq k \leq d-1$. The new scheme requires $2d$ time units to achieve $(d-1)$-fault-tolerant broadcast in a d-dimensional hypercube. Thus, it is optimal in terms of the number of time units required for the whispering model. Furthermore, for a non-faulty d-dimensional hypercube, only $2d-1$ time units are sufficient for the last time unit in phase two can be omitted entirely.

For the shouting model, after a node has received the message for the broadcast phase, it can immediately start

sending out messages for the broadcast phase and the extended broadcast phase. The broadcast phase requires d time units and one more unit is required for nodes that received the message in the d^{th} time unit to send their messages for the extended broadcast phase. Hence, $(d+1)$ time units are required which is optimal for short messages.

$(d+1)$ time units are sufficient because a node v which receives the message in time unit i in the broadcast phase must have received messages from the extended broadcast phase through its links of dimension 1 to $(i-1)$ before time unit $(i+1)$. Since v receives the message in time unit i, the Hamming distance between v and the originator is equal to i. Bit $(i+1)$ to bit d of the originator and v are the same. Consider a node u that sends a message to v in the extended broadcast phase through one of the links of dimension 1 to $(i-1)$. The Hamming distance between bit 1 to bit i of u and v must be equal to 1. Bit $(i+1)$ to bit d of u and v are the same. The Hamming distance between u and the originator is at most $(i-1)$. u must have started its extended broadcast before time unit $(i+1)$. Hence, v must have received all the messages in the extended broadcasting from links of dimension 1 to $(i-1)$ before it sends out its messages for the extended broadcast through links of dimension 1 to $(i-1)$. Furthermore, using the same argument as in the whispering model, d time units are sufficient for non-faulty hypercubes.

The new $(d-1)$-fault-tolerant broadcasting scheme can be modified easily to become a k-fault-tolerant broadcasting scheme where $k < d-1$. Instead of requiring d time units in the extended broadcast phase, $k+1$ time units are sufficient. The algorithm proceeds the same way as the $(d-1)$-fault-tolerant broadcasting scheme from time unit 1 to $k+1$. It is obvious that each node can receive k node disjoint calling paths in the extended phase.

3 A $(d-1)$-fault-tolerant all-to-all broadcasting scheme for d-dimensional hypercubes

The new k-fault-tolerant all-to-all broadcasting scheme is based on the k-fault-tolerant broadcasting scheme described in the last section. If each node initiates the k-fault-tolerant broadcasting scheme, a k-fault-tolerant all-to-all broadcasting scheme is achieved.

Assume that each node has enough memory to store n messages in an array M. Each node also has a two dimensional array L of integer with d rows and $n/2$ columns. The array L is used to store which node's message is received through a particular link during the first and third phase of the scheme.

The k-fault-tolerant all-to-all broadcasting scheme consists of four phases. The first two phases are the same as the two phases in the k-fault-tolerant broadcasting scheme. The initiator broadcasts its message and informs every node that it wants to start a all-to-all broadcasting. This is done by broadcasting a message with the id of the initiator and an integer l. The integer l is used to indicate an all-to-all broadcasting has been initiated and which phase the all-to-all broadcasting is in. In the first phase of the scheme, after a node u receives a message with id equals v through the link u_i, it stores v in row i of the array L. After the first two phases, every node knows that an all-to-all broadcasting has been initiated.

The third and fourth phases of the new all-to-all broadcasting scheme again are the same as the two phases in the fault-tolerant broadcasting scheme. Every node except the initiator starts a fault-tolerant broadcasting. Although $n-1$ nodes begin to broadcast simultaneously, the restriction that a node can only communicate with an adjacent node at any given time unit for the whispering model is not violated. This is possible because each node sends out its message or relays other nodes messages through links of the same dimension in any given time unit. Similar to the first two phases, messages are sent out with the id of the node that originates the message and an integer l. Moreover, after a node u receives a message in the third phase with id equals v through the link u_i, it stores v in row i of the array L. After the third and fourth phases, the k-fault-tolerant all-to-all broadcasting is completed.

A description of the four phases of the fault-tolerant all-to-all broadcasting scheme is given below. Let m_i be the message originated from node i, L_j is the set containing all the nodes stored in the j^{th} row of the array L, o is the initiator of the all-to-all broadcasting, M is the set of messages received by a node, ID_{all} is the set of id in which a node has received a message, and ID_m is the set of id received in a message.

A k-fault-tolerant all-to-all broadcasting algorithm

```
For the initiator o
Begin
    For time unit 1 to d do    (* In phase one *)
        begin
            send(m_o, ID_m = o, l = 4d - 2, o_i);
            store o in L_i;
        end;
    For time unit (d + 1) to 2d + 1    (* In phase two *)
        do nothing;
        Gossip(3, 2);
        Gossip(4, 1);
End;

For a node v
Begin
    receive(Message, ID_m, l, v_j);
    store Message in M;
    store ID_m in ID_all;
```

```
    store ID_m in L_j;

    If l > 4d then  (* received message in phase one *)
        begin
            Gossip(1, j);
            Gossip(2, 1);
        end
    else Gossip(2, j);  (* received message in phase two *)

    Gossip(3, 1);
    Gossip(4, 1);
End;
```

The all-to-all broadcasting algorithm is equivalent to running the fault-tolerant broadcasting scheme twice. The first two phases are the same as the two phases of the fault-tolerant broadcasting scheme. The last two phases are equivalent to having n nodes running the fault-tolerant broadcast scheme at the same time. At most $(nd - n + 1)$ messages are required for a node to broadcast a message using the fault-tolerant broadcasting scheme. Hence, at most $n(nd - n + 1)$ messages are required for the fault-tolerant all-to-all broadcasting scheme.

Assume that each message is of size F. Let β be the start up time and τ be the time to transmit one bit. For the whispering model, the time requirement for the first two phases is the same as the fault-tolerant broadcasting scheme. The first two phases require $2d(\beta + F\tau)$ time units. For the last two phases, the number is higher because the message sent in a given time unit can contain messages from up to $n/2$ nodes. In the first time unit of phase three, each message is of size F. In the second time unit, each message sent is of size $2F$. In the i^{th} time unit, the size of a message is $2^{i-1}F$. The total messages sent in all the time units in phase three is equal to $(2^0 + 2^1 + ... + 2^i + ... + 2^{d-1})F = (n - 1)F$.

```
Gossip(i, j) for a node v
begin
    if i = 3 then
      begin
          store m_v in M;
          store v in ID_all;
      end;

    For time unit w = (i − 1)d + j to i ∗ d do
      begin
          k = w mod d;
          l = l − 1;
          if i = 1 or i = 3 then
          begin
              Message = M;
              ID_m = ID_all;
              store ID_m in L_k;
          end
      else
          begin
              Message = M \ m_s where s ∈ L_k;
              ID_m = ID_all \ L_k;
          end;

      send(Message, ID_m, l, v_k);

      If a message is received
          begin
              receive(Message, ID_m, l, v_k);
              store Message in M;
              if i is odd then
                store ID_m in L_k;
          end
    end;
end;
```

$(n - 1)F$. The total propagation time is at most $d(n - 1)F\tau$. However, using the same argument as in the fault-tolerant broadcasting scheme, the propagation time can be reduced by $2(n - 1)F\tau$. This is done by not sending messages that have already been sent in phase three through the same link. This reduces the propagation time by $(n-1)F\tau$. In the algorithm, this is accomplished by storing the id of the messages sent through link of dimension i in L_i. Before a message is being sent in phase four, the messages that have already been sent are removed using the information stored in L.

Similarly, a further $(n - 1)F\tau$ is reduced by not sending messages through a link where a node receives those messages. In the algorithm, this is accomplished by storing the id of the messages received through link of dimension i in L_i. Again messages that have already been received through the link of dimension i are removed before a node sends through the same link. The total propagation time for phase four becomes $(n - 1)(d - 2)F\tau$. Hence, the total time required for the entire fault-tolerant all-to-all broadcasting scheme is $4d\beta + 2dF\tau + (n - 1)(d - 1)F\tau$. Furthermore, for non-faulty hypercubes, the time required is $(4d-1)\beta+2dF\tau+(n-1)(d-1)F\tau$ because the last time unit of phase four can be omitted entirely.

For the shouting model, a similar algorithm can be derived. The start up time can be reduced by $(2d-2)\beta$ because only $(d + 1)\beta$ are required for each of the two phases. The total time required becomes $(2d+2)\beta+(d+1)F\tau+2(n-1)F\tau$ for faulty hypercubes. If the hypercube is non-faulty, the time required is $(2d\beta + dF\tau + (n - 1)F\tau)$.

4 Comparison

In Fraigniaud's paper[5], he assumes that all the nodes start his fault-tolerant all-to-all broadcasting algorithm simultaneously. If we make the same assumption, phase one and phase two of the new fault-tolerant all-to-all broadcasting scheme can be removed. Each node starts a fault-tolerant broadcast in phase three. For the whispering model, the total time required for the new scheme is $2d\beta + (n-1)(d-1)F\tau$ for faulty hypercubes that is $(n-1)F\tau$ faster than Fraigniaud's scheme. Furthermore, the new scheme requires $n(n-1)d - n(n-1)(d-1) = n(n-1)$ fewer message transmissions than Fraigniaud's scheme. If the hypercubes is non-faulty, the new scheme requires even less time. However, for the shouting model, the new scheme requires $(n-1)F\tau$ more time than Fraigniaud's scheme.

5 Summary

A k-fault-tolerant all-to-all broadcasting scheme is proposed for a faulty d-dimensional hypercube where $0 \leq k < d$. The new scheme requires $(n-1)F\tau$ less time and $n(n-1)$ fewer message transmissions compared to previously proposed fault-tolerant all-to-all broadcasting schemes.

References

[1] Bagchi, A., S.L. Hakimi, J. Mitchem, and E. Schmeichel, Parallel Algorithms for Gossiping by Mail, *Information Processing Letters*, volume 34, 1990, pages 197-202.

[2] Bruck, J, Optimal Broadcasting in Faulty Hypercubes via Edge-Disjoint Embedding, *Networks*.

[3] Carlsson, S., Y. Igarashi, K. Kanai, A. Lingas, K. Miura, and O. Peterson, Information Disseminating Schemes for Fault Tolerance in Hypercubes, *IEICE Trans. Fund.*, volume E75, 1992, pages 255-260.

[4] Chlebus, B., K. Diks, and A. Pelc Optimal Broadcasting in Faulty Hypercubes, *In Digests of Papers of the* $h21^{st}$ *International Symposium on Fault-Tolerant Computing*, The Computer Society, IEEE, June 1991, pages 266-273.

[5] Fraigniaud, P, Asymptotically Optimal Broadcasting and Gossiping in Faulty Hypercube Multicomputers, *IEEE Transaction on Computers*, 41(11), November 1992, pages 1410-1419.

[6] Gargano, L., Tighter Time Bounds on Fault Tolerant Broadcasting and Gossiping *Networks*, volume 22, 1992, pages 469-486.

[7] Hedetniemi, S.M., S.T. Hedetniemi, and A.L. Liestman, A Survey of Gossiping and Broadcasting in Communication Networks, *Networks*, 1988, 18, pages 319-349.

[8] Johnsson, S., C.-T. Ho, Optimal Broadcasting and Personalized Communication in Hypercubes, *IEEE Transaction on Computers*, 38(9), September 1989, pages 1249-1268.

[9] Krumme D.W., Fast Gossiping for the hypercube, *SIAM Journal on Computing*, 21(2), April 1992, pages 365-380.

[10] Petrini, F., Total Exchange on Wormhole k-Ary n-Cubes with Adaptive Routing, *Proc. of the first Merged IEEE international Parallel Processing Symp. and Sypm. of Parallel and Distributed Processing*, pages 267-271, March 1998.

[11] Peercy, M., and P. Banerjee, Distributed Algortihm for Shostest-Path, Deadlock-Free Routing and Broadcasting in Arbitrarily Faulty Hypercubes, *In Digests of Papers of the 20th International Symposium on Fault-Tolerant Computing*, The Computer Society, IEEE, June 1990, pages 218-225.

[12] Ramanathan, P., and K. G. Shin, Reliable Broadcast in Hypercube Multicomputers, *IEEE Transactions on Computers*, Dec 1988, 37(12), pages 1654-1657.

[13] Scott, D.S., Efficient All-to-All Communication Patterns in Hypercubes and Meshes Topologies, *Proc. Sixth Conference Distributed Memory Concurrent Computers*, pages 398-403, 1991.