Title	Fault tolerant all-to-all broadcast in general interconnection networks
Author(s)	Sun, Y; Cheung, PYS; Lin, X; Li, K
Citation	The 1998 International Conference on Parallel and Distributed Systems, Tainan, Taiwan, China, 14-16 December 1998. In International Conference on Parallel and Distributed Systems. Proceedings, 1998, p. 240-247
Issued Date	1998
URL	http://hdl.handle.net/10722/46137
Rights	©1998 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Fault Tolerant All-to-All Broadcast in General Interconnection Networks

*Yuzhong Sun Paul Y.S. Cheung Xiaola Lin *Keqin Li*
Department of Electric and Electronic Engineering
The Univ. of Hong Kong
Pokfulam Road, HK
{ysun, cheung, xlin}@eee.hku.hk
**Department of mathematics and Computer Science*
State University of New York, New Paltz
Lik@npvm.newpaltz.edu

Abstract

With respect to scalability and arbitrary topologies of the underlying networks in multiprogramming and multithread environment, fault tolerance in acknowledged ATAB and concurrent communications become a challenge to reliable general wormhole routing multicompter with arbitrary topologies. In this paper, virtual ring tree (VRT) is proposed to deal with the challenge. An only single startup is needed in the two proposed algorithms by a simple virtual node space, which also reduces complexity of routing at intermediate steps of ATAB algorithms and re-beginning an ATAB, by cacheable virtual channels [3]. The proposed algorithm can automatically handle static faults in networks.

1 Introduction

Irregular networks may rise in Network of Workstations, cluster, and high-performance internet computation platforms. All-to-all broadcast is one of collective communications, where each node sends the same message to all other nodes on the underlying irregular networks with arbitrary topologies. Which also is referred to general networks. Reliability of ATABs with occurrences of faults on the underlying general network requires each node should acknowledge that all other nodes receive the messages broadcast by a node in distributed database and compute-intensive applications such as FFT and matrix transposition, which is called acknowledged ATAB (AATAB). This paper focuses on techniques that realize such AATAB on the underlying faulty general wormhole routing networks with arbitrary topologies.

We proposed a fault tolerant AATAB algorithm on general wormhole routing interconnection networks, called *virtual ring tree* (VRT) algorithm. For connected general interconnection networks, it is prove that a VRT can be constructed with respect to *virtual* channels, which acts as a broadcast tree. In the past, most concerns were put on how to obtain the minimum congestion, optimal message transmission and transferred messages. In most cases of networks with irregular topologies, it is impossible to satisfy simultaneously the three objectives. The large effect of start-up at each message pass is a key fact. Utilizing the non-sensitivity to distance of wormhole routing and *virtual* channels, the VRT algorithm can minimizes the great effect of start-up in message passes. In addition, an efficient simple encoding scheme is applied to simplify the routing complexity at each step. In the VRT algorithm, the same simple self-routing scheme runs at each node without any additional address calculations of next-step destinations at each step, whose an additional benefit is to make the VRT algorithm fully distributed compared to ATAB algorithm [4].

The paper organization follows. In the Section II, some basic concepts are introduced. ATAB in cycles, the fault-tolerant VRT algorithm in arbitrary connected networks follows in the Section III, which is followed by the conclusions in the Section IV.

2 Preliminaries

All-to-all broadcast patterns in interconnection networks have been analyzed in hypercube [8]. Further, two widely used all-to-all broadcasts, index and concatenation, were studied in multiport message-passing systems [4]. In this paper, the focus is put on all-to-all broadcast or concatenation.

Definition 1 An *acknowledged* all-to-all broadcast (ATAB) is a collective communication such that each processor in a system comprised of N processors, $p_0, \ldots, p_{N-1}$, having a private data $B(p_i)$ in the length of b, sends simultaneously its $B(p_i)$ to all other processors, $0 \le i \le N-1$, while each node should acknowledge all other nodes do receive its message.

A model of a multiport fully connected message-passing system ever was adapted in [4], where the emphasis on generality and flexibility of the algorithms [4] neglects detailed establishing virtual paths for one-to-one communication. In this paper, we focus on the detailed temporal overheads incurred by startup latency and congestion. We assume a relatively loose model of a *k*-port connected wormhole routing system such that one processor can send *k* messages to *k* different destinations and simultaneously receive *k* messages from *k* different sources, where $k = d$ and *d* is denoted by the degree of nodes in the system.

Some communication models characterize communication performance including BSP model, Postal model, LogP model, and linear model. The more detailed aspects including send and receive operations and buffers do not take into account in this paper. Regarding irreugular networks, the linear model [2] is applied to estimate the communication complexity of the proposed AATAB algorithm. In the linear model, the time to send an *m*-byte message from one source to one destination, without congestion, can be modeled as $T = a_0 + m \cdot b_0$. a_0 is the startup latency in which a message of 0 byte is sent from the source to the destination while b_0 is the communication time to send an additional message of 1 byte from the source to the destination.

Contracting in graph theory is applied to construct a virtual ring tree for a connected graph *G*, whose definition follows [1]. Based on graph contracting, the other two concepts, *w-partition* and *bounded w-partition* are given [7].

The maximum data blocks a processor communicates simultaneously with its router in networks in one network cycle is determined by its port capability. By *k*-port model and acknowledge ATAB, we can directly derive the minimal network cycles required by a processor receiving all messages in an acknowledged ATAB shown in Lemma 1. Compared to optimal conditions [4], lemma 1 emphasizes the minimal storage requirement per node and minimum congestion in acknowledged ATABs in general networks.

Lemma 1 The minimal number of network cycles by an ATAB in a *k*-port connected general network with *N* nodes is $\lceil (N-1)/k \rceil$. Such an ATAB is called *port-optimal*.

Lemma 2 An ATAB in a *k*-port general network with *N* nodes is without any congestion if it is port-optimal.

3 Acknowledged All-to-All Broadcast Algorithms

3.1 Overview of Virtual ring tree Algorithms

A connected network can be taken as an undirectional graph *G=G(V,E)*, where the vertex in *V* corresponds to a processor in the network, and one edge of *E* corresponds to a physical link between two processors in the network. With respect to connectivity of *G*, a span tree *T* of *G* can be constructed. Constructing a virtual ring tree of *G* has two steps. In the first step, we find a connected bounded *w-partition* G' for *G* such that all nodes of *G* corresponding to one node of G' are connected. We assume that each bi-directional physical link in *G* has at least four *virtual* channels in each direction. In the following step, a span tree T' of G' is constructed, which is called a *virtual ring tree* (VRT) of *G*, whose one node is a virtual ring in *G*. We assume that each link in a virtual ring occupies only single virtual channel in general. The root of T' may be an arbitrary node in G'. This assumption is reasonable with respect to non-sensitivity to distance from sources to destinations in wormhole routing. The problem of selecting optimal shortest path systems with minimized maximum edge congestion in connected networks with arbitrary topologies is very complicated and may be NP-complete [5][1]. Shortest paths are not applied to the proposed ATAB algorithm.

After a VRT for *G* is constructed, all nodes in *G* are encoded to self-route in all-to-all broadcast communication on networks. The code for a node in *G* consists of the two parts, one for identifier of virtual rings, and the other one identifier of a node in a virtual ring. The VRT algorithms in connected networks work in a *hot potato* fashion. In the VRT of a general connected network, a cacheable virtual channel circuitry [3] is setup along each virtual ring during a whole all-to-all communication, respectively. A message is a *hot-potato* to a node if and only if the message is what the node wants to broadcast or receive. The basic idea of the VRT algorithm is that each node broadcasts *hot potato*es in its *virtual* ring as long as it detects the *hot potato* messages and its output virtual link is idle. Also, a fair send strategy is applied to determine one node how to add a new message in AATAB. In this paper, a communication step is the maximum time it takes for a message produced by a node to travel around the virtual ring in a virtual ring tree.

3.2 General VRT Algorithm for AATAB in General Networks

The general VRT algorithm in general networks is comprised of acknowledged all-to-all broadcasts performed in virtual rings of the VRTs of the networks. Acknowledged all-to-all broadcast in a virtual ring is described and analyzed, based on which a global acknowledged all-to-all broadcast is implemented in general networks.

3.2.1 AATAB in a Ring

When each packet requires one unit of time or slot to be transmitted over a link, the optimal time to perform an ATAB is $\lceil N-1/2 \rceil$, where N is the number of nodes of the ring [6]. The algorithm for ATAB in a ring achieves the optimal time in [6]. In some critical cases, a node must wait for acknowledged message from its destinations after sending messages. In such cases, the optimal time is N. In this paper, we assume that each node must receive an acknowledgement from its destination after sending a message to the destination in a virtual ring. Our AATAB in a ring is that each node with address j sends to a node with address $j+1$ a message [7]. The address of a destination is $(j+1)$ mod N, where N is the number of nodes in the ring. The optimal broadcast [6] and our scheme are shown in Figure 3.1.a and b, the messages are transmitted in a single constant direction in a virtual ring in our scheme while the messages can be transmitted simultaneously in two directions. More importantly, one message transmitted in a virtual ring is removed from the ring by its source node in our scheme while one message may be removed by any node which has received one copy of the message before in [6]. The advantage of our broadcast scheme is that only one virtual channel is required at one physical link, while the optimal broadcast [6] requires two virtual channels at one physical link with contrary directions. Our algorithm is shown in Figure 3.2.

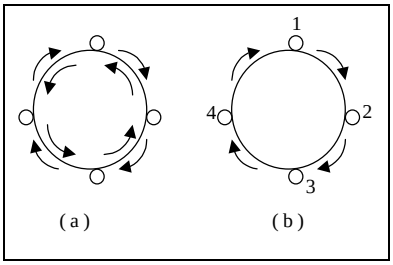

Figure 3.1 Two optimal broadcasts in a ring of 4 nodes.

Input a ring with n nodes, one message to be sent in each node
1) for k=1 to n;
2) if the node a_k with address k has a idle virtual channel; {
3) if the node a_k has any hot potatoes highest priority; {
4) if k<n;{
5) it sends one hot potato to the node with address k+1; }
6) if k=n;{
7) it sends one hot potato to the node with address (k+1)mod n;}}}
8) if a received message from other nodes was sent by a_k itself; {
9) remove the message from the virtual ring ;
10) if the received message is the same as the original message; {
11) discard the message; Goto 13) }
12) resend the message;}
13) End k.

Figure 3.2 AATAB in a ring, called *B-ring* algorithm.

Proposition 1 In a ring with $N>2$, the *B-ring* algorithm is deadlock-free, where N is the number of nodes in the ring.
Proposition 2 In a ring with the size of nodes $N>2$, an acknowledged all-to-all broadcast performs in $a_0 + b_0 \cdot \chi \cdot (N-1) \cdot \Sigma_{\max}$, where $\Sigma_{\max}$ is the maximum length of messages, and χ is the maximum number of messages produced by one node in the ring.

3.2.2 AATAB in General Connected Networks

Spanning tree schedules have been applied to broadcast communications in many interconnection networks including mesh, torus and hypercube. Broadcast communications are different from all-to-all broadcast communications. Spanning trees of connected graphs take a challenge of communication imbalance or congestion when applied to ATAB communications. The main reason is that one node wishes to transfer its messages to its parents in parallel when the destinations are not its direct children or direct father. This phenomenon shown in Figure 3.3 is called *root congestion*, which results in all messages of nodes are transmitted simultaneously to nodes closer to the root. It is obvious that the part closer to the root in a tree including the root becomes a bottleneck in such ATAB communications.

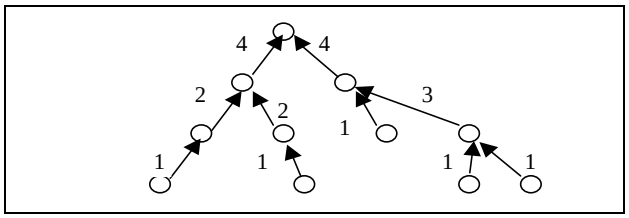

Figure 3.3 Imbalance in a tree. The weights signify the traffic while the arrows denote the traffic directions.

To solve the severe imbalance in the above ATAB schedule, some *phases* algorithms are designed based on spanning trees, where a specified subset of nodes are admitted to send their messages to some specified destinations at each phase. However, at each phase, all communication paths must be re-established. Therefore, they suffer long aggregated startups in a multithread and multitask environment.

A virtual ring tree is designed to overcome the disadvantages of ATAB performed in a single spanning tree. It is impossible or unnecessary to establish a spanning tree for each node in general networks. An alternative is to apply multiple spanning trees or disjointed pathwise spanning trees to perform ATAB on general networks to alleviate root congestion and usual edge congestion. The basic idea of a *virtual ring tree* for a network is to establish multiple virtual rings covering all nodes in the network with the maximum number of nodes in the rings w. A bounded w-partition is done in the network such that each virtual ring is reduced to a node in

the resulted *w*-partition of the network. Then, we construct multiple disjointed subtrees, whose roots are connected, in the *w*-partition by a ring shown in Figure 3.4. A node in subtrees corresponds to a virtual ring in Figure 3.4.

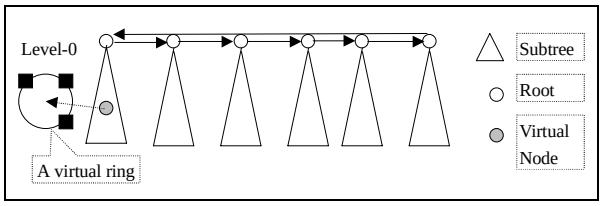

Figure 3.4 Abstract of a *virtual ring tree* of a general network.

To further alleviate root congestion incurred in all spanning subtrees, the architecture similar to one in Figure 3.4 is applied to each spanning subtrees. Through such *recursive* construction shown in Figure 3.5, root congestion and edge congestion both can be removed from all-to-all communications in general networks by such tree-based partition of general networks. The ring of roots of all spanning subtrees shown in Figure 3.4 is denoted by a *level-0 root ring (RR)*, where *j* is determined by minimal distance from the ring to the root of a general network. We assume that the level of an *RR* in general networks is zero if and only if all nodes in the ring have none of parents. For example, the level of the ring shown in Figure is zero. An example of a virtual ring tree with 5 levels is presented in Figure 3.5, where a ring is abstracted as a line with different width. Note that each level in Figure 3.5 are comprised of one or more virtual rings. For example, one of level-1 lines comprised of two virtual rings.

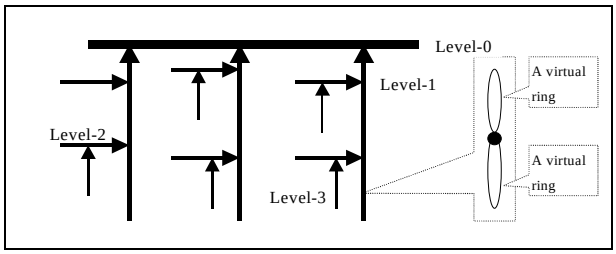

Figure 3.5 An example of the *virtual ring tree* of a general network.

Construction of a virtual ring tree is comprised of two steps in a general network. At first step, a tree similar to the one in Figure 3.5 is recursively constructed, different from the one in Figure 3.5 in that each level consists of a sequence of nodes, called a *path.* Note that each path in level-*k (k>0)* is produced by a depth-first search [32] rooted in a node in the *level-j (j=k-1)*, which performs until the first branch appears. The path of level-0 is assumed to be a diameter of a general network so as to maximize the number of levels in the VRT in a network. At the following step, each path is partitioned into a set of connected virtual rings. In wormhole routing, data flits have to suffer logic-gate delays while traversing a switch. We assume the level-*0* is a diameter of a general network. We partition the calculated level-*0* into a set of virtual rings of length *w*, identified by a level number. In a virtual ring, each node is chosen as a root to perform a depth-depth search [32]. Note that the depth-first search is stopped when a first branch is incurred. Results of all such depth-first searches in the set of virtual rings serve as level-1. In the recursive way, all level-1 paths are partitioned into sets of *virtual* trees, based on which *level-*2 *path*s are constructed. When all nodes are covered, construction of a virtual ring tree for the general network ends. The recursive construction algorithm of a VRT for a general network is shown in Figure 3.6. How to partition a *level-j* is shown in Figure 3.7. According to the algorithm, we can directly conclude the following proposition.

Input a general network, output a VRT:
1)find a diameter of the general network as a level-0; i=0;
2)work level = level-i;
3)partition the path of work level into a set of virtual rings in length w;
4)all nodes in each virtual ring in work level as a root to perform a depth-first search until the first branch is incurred;
5)all the searched paths construct a new level;
6)if all nodes are covered, stop; else i = i+1;
7)goto 3).

Figure 3.6 Algorithm of recursive construction of a VRT for general networks.

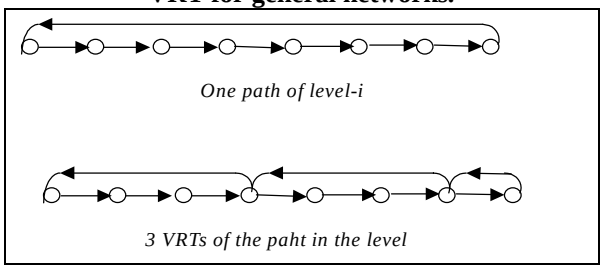

Figure 3.7 How to partition a *level.*

Proposition 3 A virtual ring tree can cover all nodes in a general connected network *G*.

Proposition 4 Algorithm in Figure 3.6 is performed in *N* steps, where *N* is the number of nodes in a network

In a general network, an acknowledged all-to-all broadcast algorithm can be performed on a virtual ring tree for the network, called the VRT algorithm. The proposed VRT algorithm in general networks consists of two phases. In the first phase, a VRT for the network is constructed, based on which a virtual space can also established over all nodes in the network. The VRT algorithm distributes a single exclusive virtual address to each virtual node. In the second phase, according to virtual addresses of virtual nodes, each virtual node tries to make a virtual circuit connected to its neighborhood

virtual node in the same virtual ring in counterclockwise direction. If virtual circuits in all virtual rings are constructed, each virtual node can broadcast messages only in its virtual ring by *B-ring* algorithm.

Input a general network and w;
1)calculate a virtual ring tree and encode all nodes;
2)establish cacheable virtual circuits for all virtual rings;
3)each virtual ring performs B-ring algorithm;
4)end when each node broadcasts its message to all other nodes.

Figure 3.8 All-to-all broadcast algorithm, called VRT algorithm.

Proposition 5 In a general network, an AATAB can be performed by the VRT algorithm without any occurrence of deadlock.

3.3 Fault Tolerance in the VRT Algorithm

Static and *dynamic* faults are taken into account on the underlying general networks during an AATAB communication. The VRT algorithm has the essential capability of handling the arbitrary number of static faults, given the underlying networks preserve connectivity in this case. To detect dynamic faults, we assume each virtual node updates periodically automatically a status bit identifying if the node is normal. That the status bit of a node is 1 means that the virtual node receives at least one message and sends one message in time it takes the node to send one message and to receive the message. In this case, the virtual node is denoted non-faulty one. Otherwise, the virtual node is specified to be faulty while the status bit is 0.

Proposition 6 In a connected network with only static faults, the VRT algorithm can tolerate all of them if and only if the faulty network is connected.

Dynamic faults are much more difficult to deal with than static faults because the VRT of the underlying faulty network has to be adjusted adaptively. A VRT for the underlying network consists of many virtual rings with the maximum node number *w*. A dynamic fault may impair a few virtual rings in the VRT while the others continue to broadcast their messages. The pattern of a fault and the mapping of the fault to virtual rings determine the number of broken virtual rings. Two patterns of faults are considered, *link* and *node* faults. A link fault is a faulty link while the *node* fault is a faulty node in the underlying network. The VRT for a network covers parts of nodes and links. A fault is called a *valid* fault if and only if the link or node it is incident on is mapped into the VRT for the underlying network. Otherwise, the fault is denoted by an *invalid* fault. The VRT algorithm tolerates invalid faults in essence. In this paper, we are concerned with only *valid* faults.

In the VRT for a normal network, a link of the network is mapped to the two or more *virtual* links of a virtual ring, which is called the *mapped* virtual rings (MVR) of the link. A valid link fault breaks all its mapped virtual rings. We assume that each node preserves the information of all its neighbors. The basic idea of fault tolerance is that each part of a broken *virtual ring* forms automatically and simultaneously a new non-fault virtual ring. By neighborhood information of nodes, the new *virtual rings* are adaptively and automatically connected to a non-fault node shown in Figure 3.9.

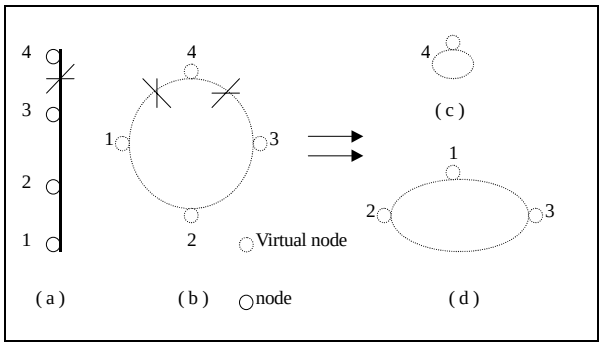

Figure 3.9 Partition a *link* faulty *virtual ring*.

We assume a virtual ring of 4 virtual nodes shown in Figure 3.9.b, whose counterpart in a link faulty network is displayed in Figure 3.9.a. The link between nodes 3 and 4 in Figure 3.9.a is mapped to the two virtual links of (1,4) and (3,4) in Figure 3.9.b. The link faulty virtual ring is broken into two parts respectively shown in Figure 3.9.c and d. The following detection algorithm is applied to construct new virtual rings. A node in the VRT for a network is a *broken virtual node* such that one of two virtual links incident to it is broken. In Figure 3.9.b, the virtual nodes 1,4 and 3 are broken virtual points.

In VRTs for non-faulty general networks, a node may be mapped to one or more virtual nodes called the *mapped virtual* nodes of the node. In Figure 3.11, how a valid faulty node partitions a virtual ring by its mapped virtual node is described. A valid faulty node partitions mapped virtual rings of the fault node into separate parts. Compared to Figure 3.9 and 3.13, valid faulty links and nodes have clearly the same effect on normal virtual rings. We can use the same scheme to deal with them. In other words, our detection algorithm shown in Figure 3.10 can handle simultaneously the two kinds of faults and have no limitation on the number of faults, given the faulty network is connected.

Input: A network with an arbitrary topology and the VRT for the network. Dynamic faults incurs possibly during ATAB. The affected virtual rings are broken into some dependent segments.

Output: A new non-faulty VRT for the network when dynamic faults occur.

T = *timeout*, T is the maximum time that a message travels a *virtual ring* without faults. Each virtual ring has unique T.

1) For each virtual node P_j' of the node *p* in a virtual ring, in parallel {
2) If P_j' can not receive the message it sends after T {
3) *P* sends to each of its physical neighbors a require message;
4) If P_j' can not receive acknowledge messages from its two virtual neighbors in T {
5) P_j' is a *broken point*;} }
6) Determine a non-faulty virtual node of the neighbor of *P*, which is in the other virtual ring;
7) if P_j' is a *break point* and successfully find a new non-faulty neighbor in the other virtual ring {
8) P_j' sends a *find-new-node* message to all its neighborhood virtual nodes in the broken virtual ring; }
9) else { P_j' makes its neighborhood virtual node a new *broken point*;
10) P_j' is not a *broken point*; go to step 7); }

11) if P_j' is not a *broken point* and receives a *find-new-node* message { /* non-broken points Relay */
12) P_j' relays the message to its next neighbor virtual node in the virtual ring; }
13) if P_j' is a broken point and receives no its broadcast *find-new-node* message in T { /* only one node in the segment */
14) P_j' and its new neighborhood virtual node form a new virtual ring;
15) the new virtual address of P_j' is combination of the virtual address of the neighbor and 1;
16) the new virtual address of the virtual node that the neighbor virtual node is mapped to in the new virtual ring is combination of its virtual address and 0; }
17) if P_j' is a *broken point* and receives a *find-new-node* message in T {
18) P_j' broadcasts the acknowledgement message including its virtual address in the broken virtual ring;}
19) if P_j' is a *broken point* and its virtual address is less than the other *broken point* it finds {
20) P_j' calculates the number of virtual nodes in the segment and relative positions of the found virtual node and itself;
21) the virtual address of the new neighbor of P_j' is combination of the original virtual address of the neighbor and its relative position, and P_j' serves as a master;
22) the new virtual address of P_j' is combination of the original virtual address of the node and relative position;
23) P_j' sends to its virtual neighbor nodes its new virtual address and +/- bit such that + is counterclockwise and - is non-counterclockwise as an *update-address message*; }}
24) if P_j' is not a *broken point* and receives an *update-address message* {
25) P_j' increases (decrease) the relative position in the received virtual address as its new virtual address according to +/- bit ;
26) if P_j' has another neighborhood virtual nodes {
27) P_j' sends to its another virtual neighbor its new virtual address;} }
28) else { P_j' broadcasts an OK message};
29) while P_j' is a master {
30) if it sends to two neighborhood virtual node its new virtual address {
31) if it receives two OK-messages { goto 34;} }
32) if it receives one OK message {goto 34}
33) while do;
34) if P_j' is a master {
35) P_j' broadcasts a beginning message within the new virtual ring to start ATAB in the ring;}
36) End do }

Figure 3.10 Construct new *virtual rings* in a faulty *virtual ring tree*.

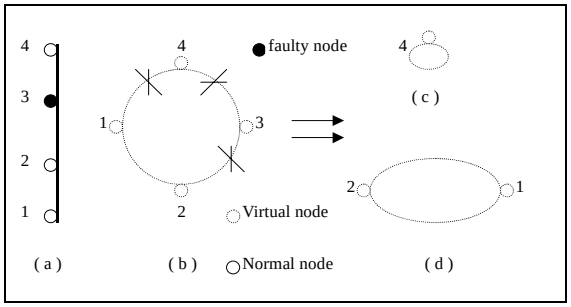

Figure 3.11 Effect of a *valid* faulty node.

Proposition 7 The detection algorithm shown in Figure 3.10 can deal with dynamic faults on general networks if and only if the faulty network is connected still.

Routing scheme in the VRT algorithm is very simple. A message from a virtual node in a virtual ring traverses only in the ring in strictly ascending order. In addition, a valid fault in a general network breaks the only its mapped virtual rings into separate segments which may not be connected to the other normal virtual rings in the faulty VRT. The VRT algorithm shows strong robustness such that all normal virtual rings can keep performing ATAB within themselves while the virtual encoding of all their virtual nodes is preserved. Nevertheless, the virtual addresses of virtual nodes in a faulty virtual ring must be changed. After new virtual rings are found by the detection algorithm in Figure 3.10, one virtual node serves as a master to re-encode all virtual nodes in a new virtual ring, which has a non-faulty neighbor node in another virtual ring. It sends an encoding message to its neighbor in counterclockwise direction. The encoding message includes its new virtual address. The neighbor

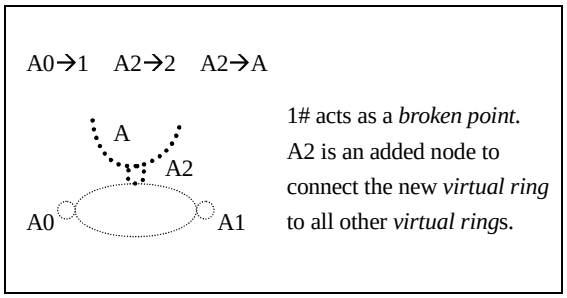

Figure 3.12 Re-encode a new *virtual ring* of Figure 3.11.d.

updates the new virtual address as its virtual address before sending its encoding message to next virtual node of the new virtual ring. After some iteration, an encoding message will return to the master. The master can make sure that the encoding is successful and sends a begin message to all virtual nodes in the new virtual ring to start ATAB in the ring. Figure 3.12 illustrates how to re-encoding all new virtual nodes in a new virtual ring. Note that no virtual nodes have the same virtual addresses in the new non-fault VRT.

3.4 Performance Analysis

An only single *virtual* channel between two adjacent *virtual* nodes is applied in a *virtual ring* of VRT for a general network. In this case, a contention for *virtual* channels on some special *virtual* nodes may be incurred, which contribute to *convergent* points. Recall that two *virtual* rings are connected by a common node divided into two different *virtual* nodes respectively in them. Therefore, we found that a *convergent* point is comprised of the only three virtual nodes located respectively on the three different adjacent *virtual* rings while they are mapped to the same node in the general network. In addition, contention may be incurred only on such *convergent* point defined as follows in Figure 3.13.

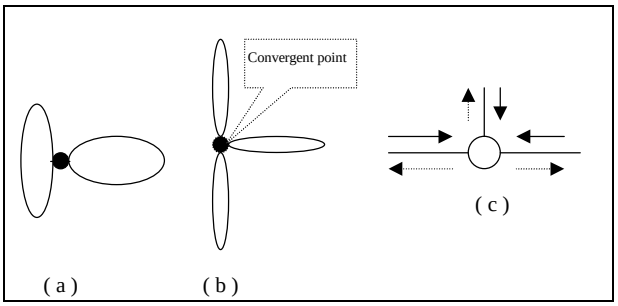

Figure 3.13 Possible contention in VRT.

Definition 2 A *virtual* node *a* is a *convergent* point of the VRT for a general network if and only if the *virtual* node has three neighborhood *virtual* nodes located in adjacent *virtual* rings of the VRT for the network.

Proposition 8 In the VRT for a general network, contention for virtual channels is incurred only on these *convergent* points of the network.

Definition 3 In the VRT for a general network, the message transmission latency from one *virtual* node P_i to any other virtual node P_j, denoted by $T_L(P_i, P_j)$, is defined to be

$$T_L(P_i, P_j) = \left(\sum_{k=1}^{M^{up}} 2D_k^{P_i} + \sum_{x=1}^{M^{down}} 2D_x^{P_j}\right) \cdot b_0 \cdot L,$$

where $D_k^{P_i}$ ($D_x^{P_j}$)is the number of virtual nodes of the *k*th (*x*th) virtual ring which is one of direct or non-direct children of one convergent point in a virtual tree along the path from P_i (P_j) to the root in the VRT, M^{up} is the number of such $D_k^{P_i}$, and M^{down} is the number of such $D_x^{P_j}$, and L is the size of transferred messages. In this paper, we assume that all nodes broadcast messages of the same length. In this paper, we assume that one network cycle is equal to $b_0 \cdot L$.

Proposition 9 The time complexity, denoted by T_Σ, of VRT algorithm in a general network can be calculated by $T_\Sigma = a_0 + \max_{i,j}[T_L(P_i, P_j)] + T_b$, where T_b is the barrier time. Note that a_0 and T_b are assumed to be determined by the underlying multicompter.

Proposition 10 In the VRT for a general network, no any contention is incurred if additional two *virtual channel*s are attached to *virtual* links incident to all *convergent* points.

Proposition 11 In a Hamiltonian network, adding an only single additional *virtual* channel to each *virtual* link incident to *convergent* points.

Proposition 12 In a general connected network (whether non-faulty or faulty), VRT algorithm transfers $L(n-1)$ units of data at most where L is the size of messages broadcast, and n is the number of nodes. In this paper, we assume one unit of data is one *byte*.

Proposition 13 For a general network, a dynamic fault can be detected and removed within $4w+2$ steps in our faulty AATAB algorithm by the detection algorithm in Figure 3.10, where w is the virtual node number of the mapped valid virtual ring of the fault.

4 Conclusions

In this paper, we discussed how to perform an acknowledged all-to-all broadcast communication in faulty and non-faulty networks with arbitrary topologies in *k*-port model. Based on an acknowledged ATAB on a ring, a fault tolerant algorithm is developed and analyzed, based on a *virtual ring tree* for the underlying network. We found that the proposed algorithm can adaptively tolerate a lot of static and dynamic faults if and only if the faulty network remains connected while the performance of ATABs may not decrease.

Acknowledgements

This work is partially supported by Hong Kong RGC f163-c and HKU CRCG grant 337/062/0012. Dr. Keqin Li is supported by the U.S. National Aeronautics and Space Administration and the Research Foundation of State University of New York through the NASA/University Joint Venture in Space Science Program under Grant NAG8-1313 (1996-1999).

References

[1] Doughlas B. West, "Introduction to Graph Theory," Prentice-Hall, 1996.

[2] P. Fraigniaud, and E. Lazard, " Methods and problems of Communication in Usual Network," *Discrete Applied Math.*, vol. 53, pp. 79-133, 1994.

[3] Binh Vien Dao, Sudhakar Yalamanchili, and Jose Duato, "Architechure Support for Reducing Communication overhead in Multiprocessor Interconnection Networks," *The proceedings of the 3rd Int. Symp. On High-Performance Computer Architecture.*

[4] Jehoshua Bruck, Ching-Tien Ho, Shlomo Kipnis, and Eli Uptal, and Derrck Weathersby, "Efficient Algorithms for All-to-All Communications in Multiport Message-Passing Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 11, Nov. 1997, pp. 1143-1156.

[5] Charles M. Fiduccia, and Paul J. Hedrick, "Edge Congestion of Shortest Path Systems for All-to-All Communication," *IEEE Trans. Parallel and Distributed Systems*, vol 8., no. 10, Oct 1997, pp.1043-1054.

[6] Emmanouel A. Varvarigos, and Dimitri P. Bertsekas, "Multinode Broadcast in Hypercubes and Rings with Randomly Distributed Length of Packets," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 2, Feb. 1993.

[7] Yuzhong Sun, Paul Y.S. Cheung, Xiaola Lin, "All-to-all broadcast in general interconnection networks," to appear in the 10th International Conference on Parallel and Distributed Computing and Systems, Las Vegas, Nevada, October 28-31, 1998.