Abstract
This paper describes an adaptive and fault-tolerant message routing technique that incorporates a distributed recovery block (DRB) scheme. The reliability requirements of inter-processor communication for parallel and distributed computer systems are increasing. The message routing algorithm, presented here, ensures the delivery of every message as long as a healthy path exists between source and destination nodes. Fault tolerance is achieved by employing distributed recovery block approach, which consists of two try-block nodes with an acceptance test. The primary try-block delivers messages, whereas the alternate is ready to take over if the primary fails. We are focusing on store-and-forward message propagation strategy, however the method can also be employed to establish a reliable path for wormhole or virtual cut through routing. The method has been investigated and implemented for 2D-mesh, torus and hypercube network topologies for varying network traffic.

Keywords: Fault-tolerant message routing, Distributed recovery block, Software and hardware fault tolerance.

1. Introduction
Inter-processor communication is an important and essential activity for parallel and distributed processing. In distributed memory parallel computers, each node is connected to a small fraction of other nodes to which it can communicate directly. Communication with non-neighboring nodes is performed indirectly via other nodes. Generally, a node consists of a processor. Memory, I/O ports but it may also have additional vector and I/O processors, VLSI routers, etc. The processes running on distinct nodes communicate with one another by message passing over interconnection links, requiring that messages be routed through other nodes. A message passing system provides a virtual connectivity of nodes and it represents a user-view of the system as if all the nodes are fully connected. Message propagation, path establishment and deadlock avoidance are the essential strategies incorporated in message passing systems. There are mainly three types of message propagation techniques namely store-and-forward, wormhole and virtual cut through. In store-and-forward, messages are sent from the source node such that each intermediate node buffers them temporarily before forwarding it to the next node along the routing path to its destination. To alleviate large message latency problem and achieve high throughput, messages are divided into smaller size packets for transmission that become the basic unit of information flow.

Inter-processor communication is an important component of system performance in multiprocessor architectures. Faults in a processor network can prevent the delivery of messages. We present a fault-tolerant routing algorithm that assures the delivery of message as long as a healthy path exists between the source and destination nodes. Adaptive routing uses alternate paths between communicating nodes providing not only resilience to failures but also makes efficient use of interconnection network bandwidth. Adaptive and fault-tolerant routing is particularly important for large-scale parallel system architecture [1]. The proposed fault-tolerant routing scheme has been implemented for 2D-mesh and hypercube processor networks and it can also be employed for other network topologies with a node connectivity of three or more. We assume that routing is done on a hop-by-hop basis with routing decisions made by intermediate nodes instead of having a path chosen by the source node. It is also assumed that each node has enough buffer space to avoid deadlock. A distributed recovery block scheme is modified for implementing the message passing system that tolerates software as well as hardware faults. The standard techniques used to cope with hardware component failures are not directly appropriate for any unanticipated situations, which result from software design faults.

A number of methods have been proposed for software systems to provide software fault tolerance. There are mainly three software fault tolerant methods including recovery block (RB), N-version programming and N self-checking programming. The recovery block paradigm essentially adapted a hardware redundancy technique (standby sparing) for use in software systems [2]. The application program is divided into subprograms or try-blocks and each block has several implementations available for execution. The primary implementation is invoked first and its execution results are evaluated by an acceptance test that verifies the authenticity of the results. If the results fail the acceptance test, the state of the process is restored and an alternative implementation is executed. Recovery block scheme is becoming popular as it can tolerate both hardware and software faults [3].
2. DRB and Message Passing

The completion time of a recovery block can be improved by executing the primary and alternate concurrently on different processing nodes of a parallel system. A parallel or distributed implementation of RB is widely known as distributed recovery block (DRB). The DRB approach is capable of minimizing the recovery time affecting forward recovery while handling both hardware and software faults in a uniform manner [4, 5].

As far as we know, DRB has not been previously employed for message routing. We have adapted the DRB scheme for message routing in processor networks with a node connectivity of three or more. For a message to be routed from a source node to a destination node, all the possible intermediate nodes between the source and destination are partitioned into overlapping 3-node DRB groups. In this way, a parallel or distributed system contains dynamically generated DRB groups where each DRB group executes a recovery block module. The output message from one DRB group becomes an input to the next group. A DRB group consists of three nodes: a predecessor or source node, a primary successor and an alternate successor node as depicted in Figure 1. The primary successor node (X) and alternate successor node (Y) has a set of try and acceptance test. Both nodes receive the same message from the predecessor or source node and apply the acceptance test (AT). The time acceptance test is used to ensure a timely message delivery by both nodes.

In a fault-free situation, primary and alternate successor nodes will pass the acceptance test (AT). The primary node notifies the alternate of its success and the message is forwarded to the next DRB group. If the primary node fails and alternate passes the AT, their roles will exchange. More specifically, the primary node (X) tries to inform the alternate node (Y) when it fails the AT. If the primary node cannot communicate its failure, the alternate node will recognize the failure upon a time out. Then the alternate takes over the role of primary node. On the other hand, if the alternate fails and primary passes the test, the primary continues with its task without being disturbed. As a retry, the alternate node requests the predecessor for another copy of the message to bring its state and local database up-to-date. Whenever a try block results in an AT failure, the node attempts to retry by getting another copy of the message from the predecessor node. The interaction between different nodes in a DRB group is essential to facilitate fault detection and recovery mechanisms.

A checkpoint may be set after the acceptance test so that each node will know whether its partner is successful or not. It is also necessary to set another checkpoint after a node forwards the message successfully. It helps the alternate node to detect the failure of its primary with a time-out. When the primary and alternate nodes are not directly connected, they can communicate through an intermediary medium (e.g. source or predecessor node).

3. Fault Recovery

An active DRB group and two successor DRB groups of a parallel system are shown in Figure 2 to illustrate the fault recovery mechanism. The active DRB group has a predecessor or source node \( P_a \), a primary successor \( P_b \) and an alternate successor \( P_c \). The primary successor DRB group has a predecessor node \( P_b \) and two successor nodes \( P_d \) and \( P_e \). The \( P_b \) also serves as a primary successor node in the current DRB group. A DRB group can have five different states depending on its normal operation and different faulty conditions.

**Fault-free Situation**

Fault-free and a primary node failure situations of a DRB group are depicted in Figure 2. The predecessor node, \( P_a \) sends the message to primary successor, \( P_b \) and alternate successor; \( P_c \) that performs their acceptance tests. When the
primary successor passes the AT, it informs its predecessor and alternate nodes about its success. If it is not a destination node, it serves as a predecessor node for the next DRB group. The alternate successor node discards its own copy of the message after receiving a successful message delivery notification from the primary. The analysis of the situation when the primary node passes and alternate fails is not so critical.

When the primary node fails just after a successful arrival of the message and during message delivery to the next DRB group, it will be reflected by the Notify Delivery status first to the predecessor and then to alternate. The alternate node will notice the abnormal case from Check Delivery status with the occurrence of a time-out event. The time-out value is a very important design parameter that should be carefully chosen. The Notify Delivery status also covers communication failures between two DRB groups.

**Primary Node Failure**

If the primary node fails the acceptance test and alternate passes, the primary and alternate exchange their roles. The alternate node immediately takes over the role of primary and delivers the message to its next DRB group as illustrated in Figure 2 by dotted lines. The new primary node notifies the old primary of a successful message delivery. Meanwhile, the predecessor node attempts to resend the message to the failed node that will retry to pass the AT. After passing the AT, the old primary node waits for the confirmation of message delivery by the new primary successor node.

**Both Primary and Alternate Nodes Fail**

This is the most critical case when both primary and alternate nodes fail their acceptance tests. To ensure recovery, the predecessor node retransmits the message to both primary and alternate nodes as shown in Figure 3. After a successful retry, both primary and alternate nodes can assume the role of primary to deliver the message to their respective successor DRB groups. However, there are two options that can be considered at this stage:

- The first option assumes that the successor node, which wakes up first, will take over the role of primary. This option provides faster recovery; however, its implementation is complex.
- In the second option, the primary node is given priority over the alternate in taking over the role of primary regardless of which node awakes first from the failure. Figure 3 illustrates the DRB operation where the initial primary node keeps its role as a primary after its recovery from a failure.

**Predecessor or Source Node Failure**

The predecessor node in a DRB group plays an important role during fault recovery. In the case of a mesh or torus topology, the primary and alternate successors use the predecessor node to exchange status information between them. When the predecessor node crashes after sending a message, the primary and alternate successors are unable to track the status of their partner. It can increase processing time in a DRB group that ends up in a situation where both primary and alternate nodes assume that their partner node has crashed. They will send identical message...
to their respective DRB successor groups and some nodes will receive multiple copies of the same message. To overcome such shortcomings, a tag is attached with each message or packet before starting its transmission from the source node. The tag identifies redundant message copies that may have been routed in the system. There are few other exceptional cases that can be avoided by the tag. For instance, if the primary node in a DRB group crashes while notifying the Message Delivery to its partner nodes, the alternate will send an identical message after the timeout.

There are some situations when the proposed message passing system may not be successful in message routing. One such case is when both primary and alternate nodes fail the AT and just at that particular time, the predecessor node also crashes. Then the predecessor node is unable to retransmit the message to its primary or alternate successor. Under these circumstances, the DRB group must be declared unhealthy for routing that particular message. The message can be retransmitted from its original source node. In this way proposed fault tolerant message routing covers 100% communication link failures but it may not cover some rare possibilities of some other failures. However, the probability of such failures is very low. There is a remote chance that both primary and alternate nodes fail together and then at that particular time the predecessor node also suffers a failure.

4. Implementation and Experimental Result

The fault tolerant message-passing system has been simulated on an IBM SP parallel computer system. Nine nodes of IBM SP can be treated as a separate RS/6000 RISC computing nodes that share data by circuit switching over a High Performance Switch (HPS). All the nine computing nodes of the system can communicate with each other directly through the HPS. A (3 x 3) 2D-array processor network is simulated on the IBM SP as shown in Figure 4. Only the adjacent nodes are allowed to exchange messages directly. For instance, if node-1 for node-5, node-1 is not allowed to send the message to node-5 directly although physically it could do. The message can only be sent to node-5 either via node-2 or node-4. In order to study the efficiency and fault-tolerant capabilities of the routing algorithm, five possible situations have been simulated for detailed investigation as depicted in Figure 5.

Six different size messages (16, 32, 64, 128, 256 and 512 bytes) are routed to analyze fault recovery and its effect on message latency for different scenarios. Node-9 acts as a source node that sends different size messages to node-1. The message latency for all the five cases (normal and failure) is provided in Figure 5. The first case is concerned with routing messages without using the DRB-base fault-tolerance. Only one successor node is selected to forward
the message. When this node receives the message, it executes the AT for authentication. This has enabled us to investigate the overhead involved due to retries of primary or alternate successor nodes. Figure 4a shows the fault-free situation of message routing path with DRB. The routing paths when primary successor and both successors fail are presented in Figure 4b and 4c respectively.

The capabilities of fault tolerance with DRB scheme are investigated by injecting faults in the alternate, primary and both successor nodes forcing them to fail. Figure 5 represents the relationship between message size and latency for routing messages from node-9 to node-1. It is observed that the message latency with DRB scheme incorporated is large as compared to routing messages without employing the DRB. The vertical gap between curve (1) and (2) i.e. the increase in message latency is attributed to DRB. The overhead is mainly due to an additional message transfer from the predecessor node to the alternate successor and the extra communication among nodes in the same DRB group. When a fault occurs in the alternate node the message latency further increases for large size messages as depicted by curve (3). The message routing path is the same as for fault-free situation but there is an extra message communication between the alternate and its predecessor for the retry. The failure of primary successor affects the system performance more than the alternate successor failure as shown by curve (4). When the primary successor fails, primary and alternate nodes exchange their roles and the alternate sends the message to its next DRB group. The worst case happens when both primary and alternate nodes fail together. In this case, the message latency is the largest among all the cases as indicated by curve (5). Curve (5) represents the situation where the initial primary resumes the role of primary after a successful retry.

In order to analyze the fault-tolerant message routing system thoroughly, a number of additional simulations were set up for different network traffic conditions. The message latency is measured in a (3 x 3) 2D-processor network for various size messages and under different traffic density.

FIGURE 5: MESSAGE SIZE VS LATENCY

In order to analyze the fault-tolerant message routing system thoroughly, a number of additional simulations were set up for different network traffic conditions. The message latency is measured in a (3 x 3) 2D-processor network for various size messages and under different traffic density.
Dummy messages are transmitted from nodes (1, 2, 5, 7 and 9) to establish the required traffic densities in the network. The destinations of messages are distributed uniformly. The message being routed from node-9 to node-1 is monitored to measure the latency for various traffic densities. Figure 6a represents message latency for the fault-free situation when both primary and alternate nodes are successful. The gap between the different curves is the incremental latency difference for the same message transfer under different traffic conditions. It is observed that this gap becomes larger with the increase in network traffic. Obviously, when the traffic gets heavier, messages take more time to reach their destinations. The latency for various failure scenarios including alternate failure, primary failure and both primary and alternate failures are presented in Figure 6b, 6c and 6d respectively.

5. Conclusion

Fault-tolerant message routing is an important and new application of the distributed recovery block approach. The routing method presented is aimed at 2D mesh, torus, hypercube and other processor networks with node connectivity of three or more. The prototype system is simulated on an IBM SP parallel system. The fault-tolerant message passing system is analyzed and investigated in detail by injecting various types of faults as well as under different traffic conditions. The results show that the DRB-based message passing is an effective way of delivering messages to intended destinations. The experimental results are encouraging as they match with the expectation on the fault coverage. The message routing is implemented for 2D mesh, torus and hypercube processor networks. The routing algorithm has some limitations also. The method uses store and forward routing strategy that has large message latencies. The work is being continued to employ the DRB-scheme for wormhole and virtual cut through routing. We believe that the method has the potential to be incorporated at least in the optimal and fault-tolerant path establishment phase of wormhole routing.

References