Destination Tag Routing Techniques Based on a State Model for the IADM Network

Darwen Rau, Jose A. B. Fortes
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Howard Jay Siegel*
Supercomputing Research Center
4380 Forbes Blvd.
Lanham, MD 20706

Abstract
A "state model" is proposed for solving the problem of routing and rerouting messages in the Inverse Augmented Data Manipulator (IADM) network. Using this model, necessary and sufficient conditions for the reroutability of messages are established, and then destination tag schemes are derived. These schemes are simpler, more efficient and require less complex hardware than previously proposed routing schemes. Two destination tag schemes are proposed. For one of the schemes, rerouting is totally transparent to the sender of the message and any blocked link of a given type can be avoided. Compared with previous works that deal with the same type of blockage, the time-space complexity is reduced from \( O(\log N) \) to \( O(1) \). For the other scheme, rerouting is possible for any type of link blockage. A universal rerouting algorithm is constructed based on the second scheme, which finds a blockage-free path for any combination of multiple blockages if there exists such a path, and indicates absence of such a path if there exists none. In addition, the state model is used to derive constructively a lower bound on the number of subgraphs which are isomorphic to the indirect Binary n-Cube network in the IADM network. This knowledge can be used to characterize properties of the IADM networks and for permutation routing in the IADM networks.

1. Introduction
This paper discusses novel and efficient techniques for routing and rerouting messages in the Inverse Augmented Data Manipulator (IADM) network [6]. These results are based on a new approach, the "state model," which characterizes and correlates the topologies of the IADM and Indirect Binary n-Cube networks, and leads to efficient exploitation of the reducibility available in the IADM network.

The class of data manipulator networks, introduced in [3], includes, among others, the Augmented Data Manipulator (ADM) network [6], the IADM network [9] and the Gamma network [13][14]. The IADM network and the ADM network differ only in that the input side of one of them corresponds to the output side of the other and vice versa. The Gamma and the IADM networks are topologically equivalent; however, they use switches of different types. Each 3x3 crossbar switch used in the Gamma network can connect simultaneously all three inputs to all three outputs whereas each switch used in the IADM network can connect only one of its inputs to one or more of its three outputs. The main interest of this paper is the study of the IADM network; both the one-to-one and permutation routings are considered. The schemes proposed for routing and rerouting messages in the IADM network are also applicable to the Gamma network.

Perhaps the most popular class of multistage networks is the multistage cube-type networks such as the Indirect Binary n-Cube [16], Omega [8], Baseline [22], Generalized Cube [20], STARAN chip [2] and a special case of SW-Banyan [4] networks. Some results of this paper are based on characteristics of the Indirect Binary n-Cube network (herein referred to as the ICube)[5].

networks. Since the cube-type networks mentioned above are all topologically equivalent [16][19][22][23], the results in this paper are also relevant to any of them.

In cube-type networks, the interchange box at stage \( s \) needs to examine the \( s \)-th bit of the binary representation of the destination address of an incoming message. If the \( s \)-th bit is \( 0 \), then the upper output of the box is taken. If the \( s \)-th bit is \( 1 \), the lower output of the box is taken. These schemes are known as destination tag routing schemes [6][16] and are extremely efficient and simple to implement. Previously proposed routing schemes [6][10][12] for the IADM network can be thought of as destination tag schemes; that is, they require calculation of the distance from source to destination in order to generate routing and rerouting tags.

McMillen and Siegel [9] proposed three dynamic rerouting techniques for the IADM network for avoiding faulty or blocked \( s \)-th (nonstrait) links. The first and the second schemes require that switches be capable of performing two's complement and \( s \)-th addition operations, respectively. The third scheme requires one extra tag bit which is dynamically updated as the message propagates toward the destination.

Parker and Raghavendra [13] used redundant number representation and proposed an algorithm capable of finding all routing paths, which, effectively, are the redundant number representations for the distance between the source and the destination. Because of the complexity of the algorithm, the cost of computation is prohibitively large so that it is infeasible to implement the algorithm in order to achieve dynamic routing [21]. Also, there is no specific work on rerouting schemes in [13][14].

Lee and Lee [7] proposed signed bit difference tag and destination tag local control algorithms for the ADM and IADM networks that require no computation for the distance between the source and the destination. But their local control algorithms can only find one routing path for each source and destination pair. If the need for rerouting arises, they still resort to the distance tag schemes to find alternate paths.

Past research has shown interesting relationships between data manipulator and cube-type networks. For example, because it is possible to embed the Generalized Cube network in the ADM network [1][16], the set of interconnections implementable by the ADM network is a superset of that of the Generalized Cube network. Analogously, the relationship applies to the IADM network and the ICube network.

Section 2 of this paper introduces a state model to describe and correlate topologies of the ICube network and the IADM network. Necessary and sufficient conditions for performing rerouting in the IADM network are derived in Section 3. In Section 4, two routing and rerouting schemes are proposed based on the theory developed in Section 3. A class of subgraphs in the IADM network that are isomorphic to the ICube network are identified in Section 5, and it is shown how to reconfigure the IADM network under certain link faults to pass the cube-admissible permutations. Finally, Section 6 summarizes the results presented in this paper.
2. State Model Descriptions for the ICube and IADM Networks

Multistage networks can be modeled as graphs by treating interchange boxes (also called switching elements) and links of the network as nodes and edges of the graph, respectively. Another equivalent graph model [1][8] results if interchange boxes are associated with edges, and links with nodes. Both models are exemplified in Figures 1 and 3 for the ICube network. The IADM network is shown in Figure 2 according to the first model. The design of switches based on both models is discussed in [11].

![Diagram of ICube network for N=8](image)

**Figure 1.** The Indirect Binary n-Cube (ICube) network for N=8 (according to the first graph model); two possible states for each box are shown (i.e., straight and exchange).

![Diagram of IADM network for N=8](image)

**Figure 2.** The IADM network for N=8 (according to the first graph model); even, and odd, switches, 0 ≤ i ≤ 2, are enclosed with bold and regular edges respectively. The solid edges (links) show the ICube subgraph.

![Diagram of ICube network for N=8](image)

**Figure 3.** The Indirect Binary n-Cube (ICube) network for N=8 (according to the second graph model).

Clearly, the ICube network in Figure 3 can be regarded as being a subgraph of the IADM network in Figure 2. Henceforth, the second model is assumed when referring to the ICube network (i.e. Figure 3) and the first model is assumed for the IADM network.

With respect to these graph models, the nodes and the edges of the graph refer to the switches and the links of the networks, respectively. The number of switches at each stage of a network is denoted \( N \) and \( n = \log_2 N \) refers to the number of stages. The switches of each stage are labeled from 0 to \( N-1 \) from the top to the bottom. Any integer \( j \) has a binary representation \( j_{n-1} \cdots j_0 \), where \( j_{n-1} \) is the most significant bit and \( n \) denotes the number of bits. The notation \( j_{p}^{p} \) means the bits of \( j \) starting at \( j_{p} \) and ending at \( j_{p} \), where \( p \leq q \). Bit \( j_{p} \) is 1's complement of bit \( j_{p} \). Throughout this paper, \( j \) and \( j_{p}^{p} \), where \( a \) is some constant, are reserved to represent labels of switches. Also modulo \( N \) arithmetic is assumed, e.g., \( j + q \) implies \( j + q \mod N \). The notation \( j \in S_1 \) is used to indicate that a switch \( j \) belongs to stage 1 and \( j \in S_{a+b} \) is used to represent a link at stage \( a + b \) joining \( j \in S_a \) and \( j \in S_b \). A sequence of switches of contiguous stages \( j \in S_{a}, j \in S_{a+1}, \ldots, j \in S_{a+b} \) is used to represent a path from \( j \in S_a \) to \( j \in S_{a+b} \).

Notation and terminology required for the characterization of network topologies and destination tag routing schemes are introduced next. Proofs of all lemmas, theorems and corollaries in this paper can be found in [17]. A switch \( j \) of stage \( i \) is an even, switch if \( j_{i} = 0 \) and an odd, switch if \( j_{i} = 1 \). Figure 2 identifies even, and odd, switches at different stages of the IADM network of size \( N=8 \). Define the functions \( \Delta C_i \) and \( \Delta C_{i+1} \) that represent connection links at stage \( i \) as:

\[
\Delta C_{i}(j_{i}^{i}) = \begin{cases} 
0 & \text{if } j_{i} \text{ is an even, switch and } s_{i}=0, \\
2 & \text{if } j_{i} \text{ is a odd, switch and } s_{i}=0, \\
2 & \text{if } j_{i} \text{ is an even, switch and } s_{i}=1, \\
0 & \text{if } j_{i} \text{ is a odd, switch and } s_{i}=1.
\end{cases}
\]

Also, define the functions \( C_{i}(j_{i}^{i}) \) and \( C_{i+1}(j_{i}^{i}) = j + \Delta C_{i}(j_{i}^{i}) \) and \( C_{i}(j_{i}^{i}) = j + \Delta C_{i}(j_{i}^{i}) \). These definitions imply the following lemma of fundamental importance to the results of this paper.

**Lemma 2.1**

\[
C_{i}(j_{i}^{i}) = j_{i}^{i} \mod 2 \quad \text{and} \quad \Delta C_{i}(j_{i}^{i}) = \Delta C_{i}(j_{i}^{i}) \] for some value of \( q_{i}^{i} \), which depends on \( j_{i}^{i} \) and \( s_{i}^{i} \).

The notation and terminology just introduced can now be used to describe the networks of interest in this paper. The following description for a network in terms of \( \Delta C_i \), \( \Delta C_{i+1} \), \( C_i \), and \( C_{i+1} \) is called the network state model.

The ICube network is composed of \( n \) stages labeled from 0 to \( n-1 \). Each stage consists of \( 2N \) links and \( N \) switches. An extra column of switches is appended at the end of the last stage as the output switches (Figure 3) and is denoted \( S_n \). A switch \( j \in S_i \) is connected to switches \( C_{i}(j_{i}^{i}) \in S_{i+1} \), for \( 0 \leq i \leq n-1 \), \( 0 \leq j \leq N-1 \), and \( s_{i} = 0 \) or \( s_{i} = 1 \). When using destination tags, switch \( j \in S_i \) routes a message to switch \( C_{i}(j_{i}^{i}) \in S_{i+1} \) where \( d_{i} \) is the \( i \)-th bit of the address of the message destination.

The IADM network is composed of \( n \) stages labeled from 0 to \( n-1 \). Each stage consists of a column of \( N \) switches and \( 2N \) connection links. An extra column of switches is appended at the end of the last stage as the output switches and is denoted \( S_n \). A switch \( j \in S_i \) is connected to switches \( C_{i}(j_{i}^{i}) \in S_{i+1} \) and \( C_{i}(j_{i}^{i}) \in S_{i+1} \), for \( 0 \leq i \leq n-1 \), \( 0 \leq j \leq N-1 \), and \( s_{i} = 0 \) or \( s_{i} = 1 \). In other words, three links connect a switch \( j \in S_i \) to the switches \( (j-2)^{i} \), \( j \), and \( (j+2)^{i} \) at stage \( i+1 \). Sometimes \( (j-2)^{i} \) and \( (j+2)^{i} \) are used to represent links \( (j \in S_i , (j-2)^{i} \in S_{i+1} ) \) and \( (j \in S_i , (j-2)^{i} \in S_{i+1} ) \), respectively. The terms a straight link refers to link \( (j \in S_i , j \in S_{i+1} ) \) and a nonstraight link refers to links \( (j \in S_i , (j+2)^{i} \in S_{i+1} ) \) or \( (j \in S_i , (j-2)^{i} \in S_{i+1} ) \).
According to the model, two types of switches, even, and odd, are required in the IADM and ICube networks. Figure 4 illustrates the connection links of a pair of even, and odd, switches for an ICube and an IADM network of size $N 	imes N$. The $\Delta C_i$ function describes the ICube connections. For the IADM network, the connection links can be described by the union of the functions $\Delta C_i$ and $\Delta C_i$. In practice, even, and odd, switches can be identical and easily programmed (at power-up or system configuration time) to behave differently.

There are two possible routing behaviors (or states) for each switch in an IADM network. A switch is said to be in state $C$ if the routing is decided in accordance with the function $C(j,4)$ and it is in the state $\bar{C}$ if the function $\bar{C}(j,4)$ applies. On the whole, the link on which a message is routed depends on whether the switch is an even, or odd, switch, in state $C$ or $\bar{C}$, and the value of tag bit $t_2$. Also the term state of the network is used to denote collectively the states of all switches in the network.

2. Theory behind the State-Based Destination Tag Routing Schemes

It is clear that when every switch in the IADM network is in state $C$, the IADM network behaves like an ICube network and, therefore, the destination address $d_{j,0,n}$ can be used as a routing tag, i.e. $t_0 = d$. More generally, the following theorem holds.

**Theorem 3.1** Let $d = d_{j,0,n}$ be the destination in the IADM network to which a message is to be sent. Then $t = d_{j,0,n}$ is the unique destination routing tag to the destination $d$ regardless of state of the IADM network.

It is implicit in the reasoning underlying Theorem 3.1 that any link on a given path results from the appropriate choice of the state of the corresponding switch, i.e. the use of "link" $\Delta C_i(j,4)$ results from setting $j \in S_0$ to state $C$ and the use of "link" $\Delta C_i(j,4)$ results from setting $j \in S_0$ to state $\bar{C}$. Thus, given a path to the destination $d$, there is at least one network state for which the use of $d$ as the destination tag results in the routing of a message through that path.

The implication of Theorem 3.1 is that the use of a state model for the IADM network reduces the problem of finding alternate routing paths to that of controlling the states of the switches in the network. Capitalising on this idea, the following theorems show how alternate routing paths can be found in order to evade blockages in the network. A straight link blockage occurs if a straight link on the routing path is faulty or busy. A nonstraight link blockage is defined analogously. The third type of blockage, called double nonstraight link blockage, occurs if both nonstraight output links of a switch in the routing path are faulty or busy. A switch blockage occurs if the switch itself is busy or faulty. A switch blockage has the same effect as blocking all of the switch's input links and can be transformed into a link blockages problem accordingly. The discussion on rerouting in this paper is concerned only with link blockages.

**Theorem 3.2** In the IADM network, a change of the state of switch $j \in S_0$ results in a different routing path to a destination $d$ if and only if a nonstraight output link of $j$ is used on the original routing path to $d$. Moreover, the other nonstraight output link of $j$ is used on the new path.

With regard to the rerouting schemes proposed in this paper, the implications of Theorem 3.2 are twofold. First, the "if" part of the theorem implies that dynamic rerouting for a nonstraight link blockage can be achieved by changing the state of the switch whose output is the nonstraight link, which is equivalent to rerouting the message through the oppositely signed nonstraight link connected to the same switch. Thus, the same subset of destinations is reachable from the two switches whose input links are the two oppositely signed nonstraight links. Second, the "only if" part of the theorem implies that dynamic rerouting for a straight link blockage is impossible. This is true in general since every routing path in the IADM network can be the result of setting the network to some state. The only resort, if any at all, to bypass the straight link blockage is to backtrack to a switch connected to a nonstraight link on the routing path at some preceding stage and to reroute from that switch. It remains to show that an alternate routing path always exists, provided that such a nonstraight link exists. In fact, the existence of an alternate routing path partly results from Theorem 3.2, as stated in the next theorem. Figure 5 illustrates the situation in Theorem 3.3.
Theorem 3.3 Consider a routing path in the IADM network to a destination $d$ that contains a blocked straight link at stage $i$. There exists at least one network state which results in an alternate routing path that avoids the same straight link failure. However, the "only if" part of the theorem also implies that, in addition, it is not possible to devise a new rerouting scheme capable of avoiding a backtracking (or look-ahead) mechanism in order to deal with straight link blockages.

From Theorem 3.2, for a given source/destination pair, if the straight output link of a switch is on some routing path, both nonstraight output links of the switch cannot be used for routing; if one of the nonstraight output links of a switch is on some routing path, the other nonstraight link of the switch is also on another routing path and the straight link of the switch cannot be used for routing. So, for a given switch, the output link blockages that affect paths from a given source to a given destination can only be (a) a nonstraight link blockage, (b) a straight link blockage or (c) a double nonstraight link blockage. Theorem 3.3 can be used to avoid case (a) a nonstraight link blockage and Theorem 3.2 can be used to avoid case (b) a straight link blockage. If case (c) occurs, then both nonstraight link blockages of a switch are bypassed by complementing the 6th state bit while the destination bit is also bypassed due to the type of blockage. The adapted backtracking scheme is based on Theorem 3.4.

Theorem 3.4 Consider a routing path in the IADM network to a destination $d$ that contains a switch at stage $i$ whose both nonstraight output links are blocked. There exists at least one network state which results in an alternate routing path that avoids the same blocked nonstraight links at stage $i$ if and only if the original routing path to $d$ contains a nonstraight link at stage $i - k$ for some $k, i \geq k > 0$.

4. State-Based Routing and Rerouting Schemes

As mentioned earlier, the novelty of the ideas in this paper lies in the state model of the routing behavior of each switch. In previously proposed approaches, routing is determined solely by tag bits. According to the state model, the switching action of each network element is conceptually determined by its relative position (i.e., an even or an odd switch), its state (i.e., $C$ or $C'$) and a destination tag bit (i.e., 0 or 1) (Figure 4). This conceptual separation of routing information makes it possible to devise the simple routing schemes described in this section.

In the first scheme, each switch is initially set up to behave as an odd, or even, switch. In addition, each switch can dynamically be set to one of the logical states $C$ or $C'$. In other words, this scheme corresponds to a direct implementation of the conceptual view of switch states. Destination tags are used and, according to Theorem 3.1, the state of the network is transparent to the sender of the message since it only affects the path of the message and not its destination. Consequently, rerouting is also transparent in the sense that it results from a change in the node state of the network. This scheme is called the Self-Repairing State-Based Destination Tag (SSDT) scheme.

The proposed SSDT scheme has the advantages that it uses simple n-bit destination tags and is capable of rerouting messages when blockages occur in nonstraight links. For a given destination tag, the routing behavior of each switch on a possible path is determined by the state of the switch, i.e., the SSDT scheme is fully distributed and rerouting is done dynamically. Each switch requires a negligible amount of extra hardware for the detection of blocked links and the representation of two states.

The second scheme is called the Two-Bit State-Based Destination Tag (TSDT) scheme and it uses 2n-bit routing tags, which specify both the destination of the message and the states of switches on the corresponding path. The TSDT scheme has the advantage that rerouting is possible when blockages occur for straight as well as nonstraight links.

As with the first scheme, the TSDT scheme assumes that each switch is appropriately initialized to behave as an odd, or even, switch. Each "digit" of the routing tag is represented by two bits $b_i$ and $b_j$ called the state bit and the destination bit, respectively. For this scheme, the state of a switch at stage $i$ is specified by $b_{i+1} = i$ if $b_{i+1} = 0$, the switch is in state $C$ and if $b_{i+1} = 1$, the switch is in state $C'$. For all $i, 0 \leq i \leq n-1, b_i = d_i$. In general, if $j$ is an even, switch, $b_{j+1} = 00$ and $b_{j+1} = 01$ direct the message through a straight link, $b_{j+1} = 10$ through link $+2'$ and $b_{j+1} = 11$ through link $-2'$; if $j$ is an odd, switch, $b_{j+1} = 00$ and $b_{j+1} = 11$ direct the message through a straight link, $b_{j+1} = 01$ through link $+2'$ and $b_{j+1} = 00$ through link $-2'$. In general, given a switch, the destination bit specifies use of a straight link or a nonstraight link while the state bit determines the choice of the positive or the negative link (if the chosen link is a nonstraight link). Since state information is carried by the routing tag, switches are not required to determine and remember their own states, i.e., the design of the switches does not need to implement the logic states $C$ and $C'$.

From Theorem 3.2, a nonstraight link blockage at stage $i$ can be bypassed conveniently by complementing the $i$th switch while the destination bit remains the same. The TSDT scheme is restated in terms of the TSDT scheme as Corollary 4.1 below.

Corollary 4.1 Let $b_{i+1} = i$ and $b_{j+1} = j$ be the state bits of the routing tag and the rerouting tag, respectively, for the IADM network. In order to bypass a nonstraight link blockage at stage $i$, the state bit $b_{i+1}$ needs to be changed to $b_{i+1}'$. That is, $b_{i+1}' = b_i + 1$ or $b_{i+1}' = b_i - 1$.

Figure 6 illustrates an example of routing from a source to destination $d = 0$ in an IADM network of size $N = 8$. Let $b_{12} = 000000$ be the routing tag and $b_{12}$ and $b_{12}'$ denote the rerouting tags. The original tag $b_{12} = 000000$ specifies the path $(1C8S_0, 0C8S_0, 0C8S_0, 0C8S_0, 2C8S_0)$. If $(1C8S_0, 0C8S_0)$ is blocked, the rerouting tag $b_{12}' = 000100$ is obtained by complementing $b_{12}$ and link $(1C8S_0, 2C8S_0)$ is used for rerouting. This tag specifies the path $(1C8S_0, 2C8S_0, 0C8S_0, 0C8S_0, 0C8S_0)$. If $(2C8S_0, 0C8S_0)$ is also blocked, the rerouting tag $b_{12}' = 000110$ results from complementing $b_{12}$ and link $(2C8S_0, 0C8S_0, 0C8S_0)$ is used for rerouting. This tag specifies the path $(1C8S_0, 2C8S_0, 0C8S_0, 0C8S_0)$. 

Figure 6. All routing paths from $1C8S_0$ to $0C8S_0$ in an IADM network of size $N = 8$. 

321
As discussed in Section 3, a straight link blockage and a double nonstraight link blockage cannot be overcome easily; implementing a backtracking (or look-ahead) mechanism is a must in order to evade these types of blockages. Since all links in the routing path from stage 1 to stage i-1 consist of only straight links, backtracking of at least k stages is required to find the switch from which an alternate routing path branches. That is, at least k state bits need to be considered for change. Due to the similarity between Theorems 3.3 and 3.4, the TSDT schemes for finding the rerouting paths from Theorems 3.3 and 3.4 are exactly the same, as stated in Corollary 4.2.

Corollary 4.2 Let \(b_{v_1}\) - \(b_{v_n}\) be the state bits of the routing tag and the rerouting tag, respectively, for a source/destination pair in the IADM network. Let \(i - k\) be the largest stage number for \(i > k > 0\) such that a switch at stage \(i - k\) is connected to a nonstraight link on the routing path. In order to bypass a straight link blockage or a double nonstraight link blockage at stage \(i\), only state bits \(b_{v_i} \rightarrow b_{v_{i+k-1}}\) need to be changed; (i) \(b_{v_i} \rightarrow b_{v_{i+k-1}} = b_{v_i} \rightarrow b_{v_{i+k-1}}\), if the nonstraight link at stage \(i - k\) of the original path is link \(-2^-4\) and (ii) \(b_{v_i} \rightarrow b_{v_{i+k-1}} = b_{v_i} \rightarrow b_{v_{i+k-1}}\), if the nonstraight link at stage \(i - k\) of the original path is link \(-2^4\). The state bits \(b_{v_i} \rightarrow b_{v_{i+k-1}}\) have arbitrary values in both cases.

The example in Figure 6 can be used to illustrate the TSDT scheme for (a) a straight link blockage and (b) a double nonstraight link blockage. (a) Again the tag \(b_{v_i} = 000000000\) specifies a path \((1E_8, 0E_8, 0E_8, 0E_8, 0E_8, 0E_8, 0E_8, 0E_8)\). If the straight link \((0E_8, 0E_8, 0E_8)\) is blocked, the rerouting tag can be 0001100 which specifies a path \((1E_8, 2E_8, 1E_8, 0E_8, 0E_8, 0E_8, 0E_8, 0E_8)\) by having \(b_{v_i} = 1 = 0, b_{v_{i-1}} = 0\). Since state bits \(b_{v_i} \rightarrow b_{v_{i+k-1}}\) can be arbitrary, 000100, for example, is also a valid rerouting tag; it specifies a path \((1E_8, 0E_8, 0E_8, 0E_8)\). (b) The tag \(b_{v_i} = 0001100\) specifies a path \((1E_8, 0E_8, 0E_8, 0E_8, 0E_8)\). Both nonstraight output links \((4E_8, 2E_8, 1E_8, 0E_8, 0E_8)\) and \(b_{v_i} = 1 = 0, b_{v_{i-1}} = 0\) which specifies a path \((1E_8, 2E_8, 1E_8, 0E_8, 0E_8, 0E_8, 0E_8, 0E_8)\) by having \(b_{v_i} = 0, b_{v_{i+k-1}} = 0\). Since state bits \(b_{v_i} \rightarrow b_{v_{i+k-1}}\) can be arbitrary, 000100, is also a valid rerouting tag that specifies the same path.

The rerouting path computed from Corollary 4.2 is blockage-free from stage 0 to stage 1. While the rerouting path is different from the original routing path from stage 1 to stage i, the routing path from stage 0 to 1 - k remains the same. This results from the fact that backtracking always proceeds backward along the original path until it stops at stage 1 - k, and the rerouting path only changes course from stage 1 - k onwards. Although state bits \(b_{v_i} \rightarrow b_{v_{i+k-1}}\) remain unchanged, the routing path from stage 1 to n - 1 may still be altered due to the changes from stage 1 - k to 1. For example, in Figure 5, the switch on the original routing path at stage 1 is 1, whereas the switch on the rerouting path at stage 1 is 0, which may further induce changes at higher-order stages.

In the TSDT scheme, the tag can be computed by the message sender, which is assumed to know the location of faulty links and switches in the network. Thus, rerouting is transparent to the switches in the network. The tag computed by the sender of the message simply avoids the usage of faulty links and switches. Therefore switches do not require any extra hardware for rerouting purposes. An alternative is to implement dynamic rerouting for the TSDT scheme. Since backtracking is indispensable for avoiding a straight link blockage, it is required that each switch can detect the inaccessibility of any output port (connected to a switch at the next stage) and signal the presence of the blockage to the switches of previous stages [10][12]. Whether rerouting is done by the sender or dynamically is an implementation decision which depends on how many stages of backtracking are allowed. When the sender computes the tag, it must be able to identify and track the switches and links on the corresponding routing and rerouting paths (the next paragraphs explain how this is done). If any of the switches or links in the path is known to the sender as being faulty, then the sender computes another tag by changing the state bits as described in Section 5.

Locating the switches on the routing path is straightforward. For a given source and a destination d, the initial routing path can be specified by setting state bits \(b_{v_1} = 0, b_{v_{i+k-1}} = 1\) (a string of n '0's), equivalent to setting every switch in the IADM network to state C. Then every switch on the original path has label \(d_{v_1} = d_{v_{i+k-1}} = 0, 0 \leq i \leq n - 1\), as now the IADM network functions like an ICube network [8][14].

To find the switches on the rerouting path, let \(j\) be the switch whose output link is blocked. First consider the case where the blocked link is a nonstraight link. It may be an (a) positive or (b) negative link. In case (a) the switch at stage i - 1 reached by the positive link is \((j + 2)^{ES}_{i+1}\), and, from Corollary 4.2, rerouting can be done through switch \((j - 2)^{ES}_{i+1}\). Case (b) can be dealt with similarly. Let the switch at stage i - 1 on the rerouting path be \(w_{v_{j+1}}\). Then the state bits \(b_{v_{j+1}} = 1\) remain intact (equal to '0') because it corresponds to having every switch from stage i - 1 to n - 1 remain in state C so that the IADM network from stage i - 1 to n - 1 can emulate the ICube network from stage i - 1 to n - 1. Thus, bits 1 to n - 1 of the label of a switch on the rerouting path at stage 1, \(i + 1 \leq i \leq n - 1\), is \(w_{v_{j+1}}\). From Lemma 2.1, bits 0 to \(i - 1, 1 \leq i \leq i + 1\), of the label of a switch on a path to destination \(d_{v_{j+1}}\) must be \(d_{v_{j+1}}\). Hence the switch on the rerouting path from stage i - 1 to n - 1 has label \(d_{v_{j+1}} = d_{v_{j+1}}\).

Next consider the case where the blockage is a straight link or a double nonstraight link blockage so that backtracking is necessary. There are two sub-cases for each type of blockage; (i) the nonstraight link found in backtracking is a negative link and (ii) it is a positive link. Here only sub-case (i) of the straight link blockage is considered; the other cases can be dealt with similarly. From Figure 5, the switch on the rerouting path is \((j + 2)^{ES}_{i+1}, 1 \leq i \leq 1\) (a formal proof can be found in [17]). The switch of stage i - 1 on the rerouting path is \(j^{ES}_{i+1}\), if \(b_{v_1} = 0\) and \(j^{ES}_{i+1}\) is an odd switch or if \(b_{v_{i+1}} = 1\) and \(j^{ES}_{i+1}\) is an even, switch, and is \((j + 2)^{ES}_{i+1}\), if \(b_{v_1} = 0\) and \(j^{ES}_{i+1}\) is an even, switch or if \(b_{v_{i+1}} = 1\) and \(j^{ES}_{i+1}\) is an odd, switch. The identification of switches on the rerouting path from stage 1 to n - 1 is done as in the case of a nonstraight link blockage described above.

The blocked link can be represented by the two switches joined by the link. Since every switch on the original routing path and the rerouting paths can be easily identified as described above, it can be readily determined whether or not the blocked link is on the current path.

The TSDT scheme can be applied to not only one instance of some blockage but also every new blockage encountered as the message propagates along. The backtracking schemes proposed in Corollary 4.2 find a rerouting path for a straight link blockage and a double nonstraight link blockage. Nevertheless, it is possible that blockages also exist on the rerouting path; then further backtracking to a lower-order stage is needed. Since this phenomenon can recur, automated backtracking may be necessary due to blockages on the rerouting paths. In fact, based on the iterated backtracking schemes, it is derived in [17] a universal rerouting algorithm that is capable of rerouting messages when multiple blockages exist in the network. The algorithm finds a blockage-free path for any combination of multiple blockages if there exists such a path and indicates absence of such a path if there exists none.

In summary, for both SDT schemes, the binary representation of the destination address can be used directly as the routing tag. In the SDT scheme, rerouting tags result from simple bit complementing operations. In terms of complexity of the computation for a rerouting tag, the SDT scheme and the TSDT scheme for one instance of nonstraight link blockage require time-complexity O(1); an improvement over previous proposed schemes [9].

322
dealing with rerouting for a nonstraight link blockage that require time-space complexity $O(\log N)$. In [10] a single-stage look-ahead scheme for rerouting of a straight link blockage was proposed; it requires use of two's complement to compute the positive and negative dominant tags so that the scheme has time-space complexity of $O(\log N)$. Note that the single-stage look-ahead rerouting scheme is valid only for some cases of the straight link blockage; it cannot be applied to any case of the straight link blockage. From Corollary 4.2, k-stage backtracking is needed for a straight link blockage and k bits of the state bits needs to be changed; thus the complexity of the TSĐT scheme for a nonstraight link is $O(k)$. If only single-stage backtracking (corresponds to 2's look-ahead) is necessary, rerouting can be done dynamically and the complexity is $O(1)$, an improvement over the scheme in [10].

5. Permutation Routing and Cube Subgraphs of the IADM Network

The results discussed so far are a consequence of the existence of spare nonstraitk links in addition to the ICube network embedded in the IADM network. This section pursues this line further by showing that there exist multiple distinct subgraphs in the IADM network, each called a cube subgraph, that are isomorphic to the ICube network. As mentioned in the introduction of this paper, the cube-type networks have been developed extensively in the literature and shown to be topologically equivalent. Together with results from these studies, the knowledge of how to identify cube subgraphs can help the understanding of the capabilities of the IADM network and be useful for permutation routing in the IADM network.

Since each switch can be in state $C$ or $\bar{C}$, there are as many as $2^{2^n} = (N^2)^n$ network states. Setting a switch to a certain state indicates that one of its nonstraight output links can be used for routing (i.e., it is active) while the other cannot. Thus, each network state can be associated with a subgraph of the IADM network which contains only the active links. When all switches in the ICube network are set to state $C$, the IADM network functions as an ICube network; this network state corresponds to a cube subgraph. The constructive derivation of a lower bound for the number of cube subgraphs of the IADM network uses the two basic ideas discussed next.

Since $x^{-1} \equiv -x^{-1} \mod 2^n$, $C_{i,j} = C_{\bar{i},\bar{j}} = i_1(i_{n-1}) = \bar{i}_{n-1}(i_{n-1})$, i.e., the state of each switch of stage $n-1$ is irrelevant in the sense that any switch at stage $n-1$ is always connected to the same two switches at stage $n$. Consequently, given any cube subgraph, there exist $(2^n-1)$ subgraphs isomorphic to it which differ only in their choices of the nonstraight link $2^n-1$ at stage $n-1$. Therefore, the total number of distinct cube subgraphs is given by the product of $2^n$ and the number of distinct subgraphs of the IADM network from stage 0 to stage $n-2$ that are isomorphic to the same stages in the ICube network.

The calculation of the number of subgraphs in the first $n-1$ stages uses an idea similar to that proposed in [6] for reconfiguring the DR network so that it performs as a Generalised Cube network. All switches of the IADM network are logically relabeled by adding a constant $x$, $0 \leq x \leq N-1$ to the original labels, i.e., switch $j$ becomes $j = j + x$. By setting each switch to be an even, or odd, switch according to its new label and having all switches be in state $C$, a cube subgraph results for each relabeling. However, of the N possible subgraphs, only $N/2$ are distinct as far as the first $n-1$ stages are concerned. This result is stated in Theorem 6.1. A graphical interpretation of cube subgraph isomorphism for an IADM network of size $N=8$ is illustrated in Figure 7. In Figure 7, each physical switch $j$ acts as a logical switch $j = (j+1) \mod 8$. The isomorphism to the ICube network can be easily visualised by moving switch 7 to the top of each stage as shown in the figure. Notice that setting some switch to state $C$ according to its logical label may be equivalent to setting the switch to state $\bar{C}$ according its original label. For instance, switch $0 \in C_0$ (logical label 1) is set to state $C$ in Figure 7.

Theorem 6.1: There exist at least $N!N/2^n$ distinct cube subgraphs in the IADM network.

In order to reconfigure the IADM network to one of its cube subgraphs, each switch of stage $i$, for $0 \leq i \leq n-2$, needs to know the $i$-th bit of its logical label. This can be done by sending the same logical label to every switch in the same row at system reconfiguration time. Each switch is set as being an odd, or even, switch by examining the $i$-th bit of the logical label. All switches operate in state $C$ according to its logical label with the exception of those at stage $n-1$ for which different states correspond to different subgraphs.

The results of this section can be used in different ways. One usage is in characterising a class of permutations performable by the IADM network. Permutations passable by the ICube network are discussed in [16] and adaptable from [6]. Thus, the IADM network can perform all of these permutations plus the same set of permutations with a given $x$ added to both the same source and destination labels, $0 \leq x < N/2$. Another usage of the results of this section is that the IADM network can pass the permutations performable by the ICube network when the ICube network embedded in the IADM network experiences nonstraight link failures. This is done by incorporating a reconfiguration function in the system that reassigns each switch $j$ to $(i+j)$ and reconfiguring the IADM network to a corresponding cube subgraph which does not include the faulty nonstraight links. In [23] it is shown that any of the cube-type networks can pass the permutations performable by the others by incorporating appropriate reconfiguration functions. By the same token, the IADM network with a nonstraight link fault can also pass the permutations performable by the cube-type networks by including these reconfiguration functions in the system.

6. Concluding Remarks

One of the main contributions of this paper is the identification of destination tag routing schemes for the IADM network. They are simpler and more efficient than previously known approaches, thus requiring less complex switches and reducing message communication delays due to routing overhead. In the SSDT scheme rerouting can be done when nonstraight links fail and in the TSĐT scheme both the straight and double nonstraight link blockages can be avoided. As for the SSDT scheme, routing and rerouting are transparent to the source and only negligible hardware and time are used by each switch for routing and rerouting purpose. These are considerable advantages over previously proposed schemes which do not use destination tags and require extra hardware or delays of $O(\log N)$ complexity instead of $O(1)$. In addition, previous works all deal only with
certain types of blockages. Based on the TSDT scheme, a universal rerouting algorithm is derived in [17], which is capable of avoiding any combination of multiple blockages if there exist a blockage-free path and indicating absence of such a path if there exists none. The rerouting capabilities of the new schemes adequately exploit the redundancy available in the IADM network and can be readily used for fault-tolerance and load balancing purposes.

Another contribution of this paper is the constructive derivation of a lower bound on the number of cube subgraphs of the IADM network. While it was previously known that the ICube network is a subgraph of the IADM network, this paper shows that there exist at least $N^2/4$-1 distinct cube subgraphs. This, combined with previous multistage cube network studies, can help characterise some of the permutations performable by the IADM network. As other use of the subgraph analysis, it is shown how to reconfigure the IADM network under nonstraight link faults to pass the cube-admissible permutations.

Perhaps the most fundamental contribution of this paper is that of the network state model used for the IADM and the ICube networks. The essence of this model is in the recognition that the routing action of each switch is conceptually dependent on its position in the network (topological information), its state (functional information), and the destination of the message (routing information). Topological information is fixed and, when using destination tags, the same can be said of routing information for a given message destination. Consequently, the routing path is solely determined by the state of the network. These basic concepts are applicable to networks other than those considered in this paper; the state model can help devise new designs, solve routing problems, and understand relationships among networks.

Acknowledgement

For this research, (1) D. Rau and J. Fortes were supported in part by the National Science Foundation under Grant DCR-8419745, by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and was administered through the Office of Naval Research under contract No. N00014-86-k-0886, and (2) H. J. Siegel was supported by the Supercomputing Research Center under contract 8925 while at Purdue (through May 1987).

References