FAULT TOLERANCE OF DISTRIBUTED LOOPS

ABDEL AZIZ FARRAG

Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada

ABSTRACT

Distributed loops are highly regular structures that have been applied to the design of many locally distributed systems. This family of networks includes many important configurations such as rings and circulant graphs, for examples. In this paper, we examine the problem of extending a (unidirectional) distributed loop so as to tolerate any given number of node failures. We study this problem when the parameters that define the loop are given numerically (as constants) or symbolically (as variables). We demonstrate our formulation by developing (fault tolerant) solutions for daisy-chain networks. Our results show that the solutions obtained are efficient (i.e., either optimal or nearly optimal).

Keywords: *Networks, Graphs, Distributed Loops, Fault-Tolerant Solution.*

1. INTRODUCTION

Distributed loop networks have been widely used in the design of local area computer networks and also in some parallel processing systems [2,7,15]. This class of networks exhibits many useful properties, such as simplicity, expandability and regularity. Moreover, it includes (as special cases) several important topologies such as rings and circulant graphs, for examples.

Distributed loops are highly regular structures, and as a result, the failure of even one node or link can change or break the network. Therefore, to make the network fault-tolerant, some spare nodes and links can be added so that when a failure occurs, the network can be reconfigured to bypass the defective components. This is the main approach used to achieve fault-tolerance [1].

Achieving fault-tolerance by extending a given network has been examined for a variety of architectures such as (undirected) rings [6,7,9], stars [8,18], meshes [3-7,9,19] and hypercubes [3,6,10,17,19]. The main optimization criterion used in building a solution is to reduce the node degree of the (extended) network. This objective is important in practice due to the limitation on the number of links allowed per node in VLSI design [2]. This is also the same criterion used in this paper.

We examine the fault-tolerant extension problem for the distributed loop configuration. Our formulation can be used to tolerate any desired number of node failures, and moreover, it can be applied whether the parameters that define the distributed loop are stated numerically (as constants) or symbolically (as variables). For the former case, we develop a new algorithm which finds a (fault-tolerant) solution for the distributed loop G by first generating a family of solutions, and then selecting the one with the least node-degree. For the latter case, we demonstrate that the formulation can be applied analytically, and use the method to design a (fault tolerant) solution for the distributed daisy-chain network.

Some important criterion affecting the performance and the reliability of a distributed loop, (such as its diameter, connectivity and routing), have been previously examined for several special cases (see [2,11,13,14,15,16]). To the best of our knowledge, designing a fault-tolerant extension of a distributed loop has not been studied, except for the special case of a circulant graph (e.g., [7,9,10]). This paper extends and generalizes this earlier work to the (unidirectional) distributed loop.

The rest of this paper is organized as follows. First, we present the background material, and develop a new theorem for designing fault-tolerant solutions of any distributed loop (in Section 2). Then, we develop a formalism to partition the jumps of a distributed loop (in Section 3). This formalism will be used (in Section 4) to develop a new algorithm to

design a fault-tolerant solution for a distributed loop. This algorithm assumes that the parameters that define the distributed loop are given numerically (as constants). Forming solutions for cases where these parameters are stated symbolically (as variables) will be discussed in Section 5. Finally, we present conclusions (in Section 6).

2. BACKGROUND

In this section, we present the background material used throughout this paper. The (directed) graphs defined below represent multicomputer networks, where the nodes represent processors and the edges represent links.

Definition 2.1: (k-ft solution)
A graph H is a k-fault-tolerant (or k-ft) solution of a graph G, if removing any k nodes from H will leave a remainder that contains a subgraph isomorphic to G.

For example, the graph G in Figure 2.1 has a 1-ft solution H shown in Figure 2.2. (The thicker edges inside H identify a subgraph isomorphic to G which excludes one faulty node.)

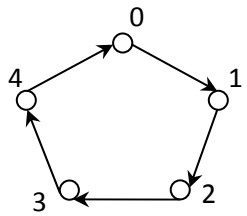

Fig. 2.1 A graph G(1:5).

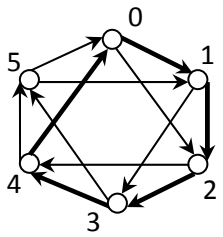

Fig. 2.2 A 1-ft of G(1:5).

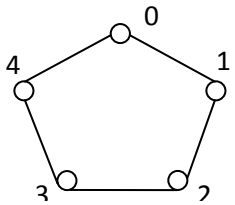

Fig. 2.3 Undirected ring.

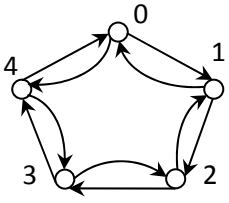

Fig. 2.4 A graph G(1,4:5).

In designing a k-ft solution of a graph G, we use the minimum number of spare nodes, i.e., the solution will have exactly k more nodes than G. Throughout the rest of the paper, G will represent a distributed loop network as defined below.

Definition 2.2: (Distributed loop)
A distributed loop is a directed graph defined by a set of nodes numbered 0,1,..,n-1 and a set of integers called jumps denoted A = $a_1, a_2, .., a_i$ such that for every node x and every jump a_j, x is adjacent to node x a_j mod n.

For example, the graph G in Figure 2.1 is a 5-node distributed loop with only one jump equal to 1. In what follows, we denote an n-node distributed loop G with jumps $\{a_1, a_2, .., a_i\}$ as $G(a_1, a_2,.., a_i : n)$. For example, the distributed loop in Figure 2.2 is G(1,2: 6).

It is convenient to arrange the nodes of the loop around a circle in a clockwise direction (as shown in Figure 2.1). We shall use only positive jumps, that is, if a negative jump a_i arises, we shall convert it into the equivalent positive jump a_i n (mod n).

Notice that distributed loops generalize rings and circulant graphs, e.g., an n-node directed ring corresponds to the distributed loop G(1: n), whereas an n-node undirected ring is equivalent to the distributed loop G(1,n-1: n) in which two directed links are used to represent each edge in the (undirected) ring as shown in Figures 2.3 and 2.4.

Checking whether a given node is adjacent to another node can be done by finding the (circulant) distance between them (along the cycle) and checking if it is equal to one of the jumps, where the (circulant) distance is defined as follows.

Definition 2.3: (Circulant-distance)
The circulant-distance from a node x to another node y is defined as either y-x or y-x+n depending on whether or not y is greater than x; respectively. For examples, in Figure 2.1the circulant-distance from node 1 to node 3 is 2 whereas the circulant-distance from node 1 to node 0 is 4.

The basic method to construct a k-ft of a distributed loop is given in the theorem below which generalizes that of [6].
Theorem 2.1: Given a graph $G(a_1,a_2,.., a_i : n)$, we can construct a distributed loop H with n+k nodes that is a k-ft of G where the jumps of H consist of the union: $a_1, a_1+1, a_1+2,.., a_1+k$ $a_2, a_2+1, a_2+2,.., a_2+k$... $a_i, a_i+1, a_i+2,.., a_i+k$.
Proof: Assume here that the number of faulty nodes is k; (otherwise, if there are fewer than k faulty nodes, we can make the difference by

selecting some extra healthy nodes and treat them as faulty). The remaining n healthy node of H will be renumbered in the same order along the circle starting at any healthy node as 0 and skipping every faulty node. That is, after this renumbering, the healthy nodes will be labeled 0,1,..,n-1.

For every healthy node newly numbered as x, let g(x) denote the node that corresponds to x before this renumbering. To complete the proof, we need only to show that for every such node x and every jump a_i, x must be adjacent to the (healthy) node $x + a_i \bmod n$.

First, assume that none of the nodes along the circle over the interval from node x up-to node $y = x + a_i \pmod n$ is faulty. Then, the circulant-distance from x to y is the same as that from g(x) to g(y), i.e., equal to a_i. Therefore, by the definition of the distributed loop H given in this theorem, node g(x) is adjacent to node g(y), which means node x must be adjacent to y.

Otherwise, let M denote the number of faulty nodes (skipped) along the circle over the interval from the node x to the node $y = x + a_i \pmod n$, where x and y are healthy nodes. In this case, the circulant-distance from node g(x) to g(y) is equal to Ma_i where $M \geq k$. Thus, by the definition of the distributed loop H, node g(x) is adjacent to node g(y), which in turn means node x is adjacent to node $y = x + a_i \pmod n$. □

For example, by the above theorem, the loop G(1,9: 11) has a 1-ft of the form H(1,2,9,10: 12). Although the above theorem finds only one k-ft for a distributed loop G, a large family of k-ft solutions can be generated as will be shown later. To compare the costs, we shall count only the number of jumps used in every solution as it relates directly to node-degree. Notice that the degree of a graph H is defined as the maximum degree of any node in H.

3. PARTITIONING THE JUMPS OF A DISTRIBUTED LOOP

A subset of the jumps of a distributed loop G is called a block if they are physically consecutive, i.e., of the form □j, j1, j2, ...□. By the (k-ft) theorem proven in the preceding section, the fewer the number of blocks in G, the more efficient the cost of its k-ft. Thus, the best-case for the k-ft constructed by this theorem occurs when the jumps of G are all consecutive (i.e., form only one block), whereas the worst-case occurs when no pair of jumps are consecutive (i.e., each block in G consists of only one jump).

In this section, we develop a new formalism called m-distance subsets which generalizes the notion of blocks. These subsets will be obtained by partitioning the jumps of the given loop as explained below.

In what follows, the greatest common divisor of two integers x and y will be denoted gcd(x,y) and the inverse of x (mod n) is denoted x^{-1}. Notice that $x.x^{-1} \pmod n = 1$. Two integers x and y are called coprime or relatively prime, if gcd(x,y)= 1.

Definition 3.1: (Partitioning sequences)
Let n and m be any pair of integers where $\gcd(n,m) = 1$ and $n > m > 0$. We define an ordered sequence, based on n and m, denoted $S(n,m) = \langle s_1, s_2, .., s_{n-1} \rangle$, where $s_i = i.m \pmod n$, for all $1 \leq i \leq n-1$.

For instance, when n=7 and m=1, S(7,1)= <1,2,3,4,5,6>. Similarly, for n=7 and m=3, we get S(7,3) = <3,6,2,5,1,4>.

It is not difficult to show that S(n,m) contains all integers from 1 up-to n-1, that is, it includes the whole range of valid jumps (of an n-node distributed loop). After generating S(n,m), we shall use it to partition the jumps of the distributed loop as will be explained below.

Definition 3.2: (m-distance subset)
A subset S of the jumps of an n-node distributed loop G is called an m-distance subset, where m is any integer such that gcd(n,m) =1 and n > m > 0, if there is a subsequence of consecutive elements in S(n,m) that contains every jump in S and has the same number of elements as in S. Further, an m-distance subset is called maximal if it is not contained in any other m-distance subset.

Example 3.1: The two jumps of the distributed loop G(1,3: 9) form a 2-distance subset. This is because S(9,2)= <2,4,6,8,1,3,5,7>, i.e., S(9,2) has a subsequence <1,3> that contains these jumps.

Definition 3.3: (m-distance partition)
Let P(A,n,m) denote a collection of m-distance subsets defined over a set A of jumps of an n-node distributed loop. Then, P(A,n,m) is called an m-distance partition of A, if every m-distance subset in P(A,n,m) is maximal and every jump in A appears (inside one subset) in P(A,n,m).

Example 3.2: Consider the distributed loop G(1,2,5:7). Since S(7,1) = <1,2,3,4,5,6>, a 1-distance partition of the jumps of G is equal to 1,2,5.

Similarly, since S(7,3) = <3,6,2,5,1,4>, a 3-distance
partition of the jumps of G has only one subset 1,2,5.

We present below the algorithm for partitioning the jumps of a distributed loop G in which A denotes the set of jumps in G, n denotes the number of nodes of G, and m is any integer that is coprime to n, where n > m > 0.

Algorithm Partition (A,n,m) {

1. Construct the sequence S(n,m) = <s_1, s_2,.., s_{n1}> as defined before;
2. For every element s_i in S(n,m), if s_i appears as a jump in A, then keep s_i in S(n,m); otherwise, replace s_i in S(n,m) by a special separation symbol, say "&";
3. For every (maximal) subsequence in S(n,m) that does not include the separation symbol "&", form an m-distance subset which has the same elements as those in the subsequence;
4. Return a partition P(A,n,m) consisting of all m-distance subsets formed above;
}

Since S(n,m) contains n-1 elements, and each of them can be checked in Step 2 in only O(log □A□) time (if A were sorted first), therefore, the time-complexity of the above algorithm is O(n log □A□).

It is not difficult to show that the sequences S(n, m) and S(n, n-m) are the reverse of each other, and therefore, if we partition the jumps of an n-node graph using S(n,m) or S(n,n-m), we obtain the same results. Accordingly, we shall use only sequences S(n,m) that satisfy the condition m < n □2.

The table below shows all possible ways to partition the jumps of the distributed loop G(1,4,7,9: 11); each of them is based on a different value of m.

Table 3.1 Every row below gives a value of m that is coprime to n, (where n= 11), the sequence S(11,m), and a partition of the jumps 1,4,7,9 corresponding to m.

M	S(11,m)	Partition of□1,4,7,9□
1	<1, 2,3,4,5,6,7,8,9,10>	□□1□, □4□, □7□, □9□□
2	<2,4,6,8,10,1,3,5,7,9>	□□1□, □4□, □7,9□□
3	<3,6,9,1,4,7,10,2,5,8>	□□1,4,7,9□□
4	<4,8,1,5,9,2,6,10,3,7>	□□1□, □4□, □7□, □9□□
5	<5,10,4,9,3,8,2,7,1,6>	□□1,7□,□4,9□□

4. THE ALGORITHM

Given a distributed loop G, we would like to construct a k-ft solution for it. This can be done by applying Theorem 2.1 directly as explained before. However, if the jumps are not consecutive, the solution obtained may be unnecessarily expensive. Therefore, to improve over this, we develop a new algorithm which works by generating a large family of k-ft solutions that can be compared to select the one with the least node-degree. This method assumes that the jumps of G are specified numerically as constants. Otherwise, if the jumps of G are given symbolically (as variables), we can use the method explained in the next section.

Theorem 4.1: If n and p are relatively prime integers, then the graph G(a_1, a_2,.., a_i : n) must be isomorphic to H(a_1.p mod n, a_2.p mod n,.., a_i.p mod n: n).

Proof: We define a mapping "f" which transforms each node x in G into a node f(x) in H such that f(x)= x.p (mod n). To prove that "f" is one-to-one, it suffices to show that "f" cannot map distinct nodes to the same value. To see why, let f(x) □f(y), then x.p (mod n) = y.p (mod n), and therefore, by multiplying each side by p^1 we get x .p. p^1 (mod n) = y.p. p^1 (mod n). But, since p. p^1 □1 (mod n), this will imply x□y.

The function "f" also preserves adjacency. This is because for any edge (x,y) in G, we must have some jump a_j such that y □x a_j (mod n).

This, in turn, implies f(y) = f(x. a_j) = x..p + a_j..p 3. (mod n),
that is, f(y) = f(x)+ a_j.p mod n. This implies (f(x),f(y)) is an edge in H.

The two properties of "f" proven above4. establishes the isomorphism between G and H. □

5. Given a distributed loop G, we can apply the above theorem to convert each m-distance subset of its jumps into a block of consecutive elements as described in the following lemma. (This is needed before we can construct a new k-ft solution for G.)

Lemma 4.1: Multiplying the jumps of the distributed loop G(a_1, a_2,.., a_i : n) by m^1 transforms each m-distance subset of its jumps into a block of consecutive integers of the form □j, j1, j2, ..□.

Proof: By the definition of an m-distance subset given earlier, the elements of any such subset can be written as □b, ḅm, ḅ2m, ...□for some integer b, where additions are mod n. Thus, if we multiply the jumps by m^1, we obtain □b.. m^1 , b.. m^1 + m.. m^1 , b.. m^1 + 2m .. m^1, ..□where the additions and the multiplications are done mod n. Since m.. m^1 = 1 (mod n), this subset can be simplified to □b.. m^1 , b.. m^1 + 1, b.. m^1 + 2, ...□. Moreover, if we substitute j = b.. m^1 , we get □j, j1, j2, ..□.

The above results, together with the formalism of m-distance partitioning given earlier, generalize the theorem of Section 2 to construct a k-ft solution as follows. Instead of finding only one k-ft solution for G, we can form many k-ft solutions; one for every integer m relatively prime to n. Thus, based on the value of m selected, we can form an m-distance partition of the jumps of G; and then convert its subsets into blocks (as was shown in the above lemma), and finally apply Theorem 2.1 to form a new k-ft solution for G. The complete details are given in the algorithm below.

In this algorithm, G denotes a given distributed loop with n nodes and k denotes the number of node failures to be tolerated.

Algorithm Fault-tolerance(G,k) □

1. Generate all integers {m_1, m_2,.., m_j□such that for each m_j we have gcd(n,m_j)□1 and 1□m_j < (n□2);
2. For every m_j generated above, find an m_j-partition of the jumps of the distributed loop G;
3. For every m_j-partition generated above, convert its subsets into blocks by multiplying its jumps by m_j^1 (mod n); and let T_j denote the distributed loop whose jumps consists of the union of these blocks;
4. Use Theorem 2.1 to construct a k-ft solution of each graph T_j defined in Step(3);
5. Compare all k-ft solutions constructed in Step(4) to select the one with the least node-degree;

□

For every m_j generated, it is not difficult to show Steps 2 to 5 of the above algorithm can be done in O(n log□A□k □A□). Thus, since the number of m_j's generated is bounded by O(n), the whole algorithm requires O(n^2 log□A□n k □A□).

Example 4.1: Let G =(1,4,7,9: 11) and k = 1. Then, we can use Theorem 2.1 to form a 1-ft solution for G equal to H(1,2,4,5,7,8,9,10: 12). However, if we apply the above algorithm to G, many 1-ft solutions will be generated; and out of them the solution H(3,4,5,6,7: 12) will be selected as the one with the least node degree. (The table below provides a list of every solution generated, and the value of m corresponding to it.)

Table 4.1 This table traces the various 1-ft solutions of G(1,4,7,9: 11) generated during the execution of above algorithm. The final solution selected is given in bold.

m	1-ft solution of G(1,4,7,9:11)
1	H(1,2,4,5,7,8,9,10: 12)
2	H(2,3,6,7,9,10,11: 12)
3	**H(3,4,5,6,7: 12) □selected solution**
4	H(1,2,3,4,5,6,10,11: 12)
5	H(3,4,5,8,9,10:12)

Lemma 4.2: Algorithm Fault-tolerance(G,k) runs in O(n^2 log□A□n k □A□) time.

Proof: We trace the steps of the algorithm for each m_j generated. Checking if m_j and n are relatively prime in Step(1) can be done in O($\log^3$ n) time as shown in reference [12], and finding a partition corresponding to m_j in Step(2) requires O(n log□A□) time as was shown earlier. Converting the jumps into blocks in Step(3) needs O(□A□) time, and constructing a k-ft in Step(4) requires at most O(k□A□) time. Thus, the time for each m_j is O(n log□A□k □A□), and since the number of integers (m_j's) to be generated is bounded by O(n), therefore,

the whole algorithm can be done in $O(n^2 \log A \cdot n k A)$ time. □

The following table gives several examples of graphs, as well as their 1-ft and 2-ft solutions that are computed by the above algorithm.

Table 4.2 Every row gives a graph G, its 1-ft and 2-ft.

Graph	1-ft of G	2-ft of G
G(1:5)	H(1,2:6)	H(1,2,3:7)
G(1,6:7)	H(3,4,5:8)	H(3,4,5,6:9)
G(1,3:7)	H(4,5,6:8)	H(4,5,6,7:9)
G(1,3:9)	H(5,6,7: 10)	H(5,6,7,8:11)
G(1,4,7,9:11)	H(3,4,5,6,7:12)	H(3,4,5,6,7,8:13)

5. FAULT TOLERANCE OF A DISTRIBUTED LOOP

The algorithm given above for designing a k-ft solution of a distributed loop G is simple to implement; however, it assumes that the parameters of G are given numerically, i.e., the size n and the jumps of G must be constants. If these parameters are given symbolically (as variables), a k-ft solution can be developed analytically, by applying our formulation (of m-distance partitioning) to G. The method is explained below by developing a solution for the (daisy-chain) network G(1, 2 : n), which was introduced originally in [11].

Lemma 5.1: Suppose that n is divisible by 3. Then, either n/3 -1, or n/3 +1, or both must be coprime to n.

Proof: Since every pair of consecutive integers is coprime, then each of the two pairs (n/3 -1, n/3) and (n/3, n/3 +1) must be coprime. This, in turn, implies that "3" is the only possible common factor between the pair (n/3 -1, n), and between the pair (n, n/3 +1). However, since exactly one out of any 3 consecutive integers (e.g., n/3 -1, n/3, n/3 +1) is divisible by 3, therefore, at least one of the two pairs (n/3 -1, n) or (n, n/3 +1) must be coprime. □

Lemma 5.2: Let n be divisible by 3, and let n/3 -1 be coprime to n. Then, the inverse of n/3 -1 (mod n) is equal to either (2n/3 -1) or (n/3 -1) depending on whether or not n is divisible by 9; respectively.

Proof: Since 3 divides n and n/3 -1 is coprime to n, then n/3 -1 cannot be divisible by 3. Consequently, either n/3 or n/3 +1 is divisible by 3.

Suppose first that n/3 is divisible by 3, i.e., n is divisible by 9. Then, (n/3 -1)(2n/3 -1) = 2n(n/9)-n+1. Since n/9 is an integer in this case, therefore 2n(n/9) – n +1 (mod n) = 0 -0 +1 =1. Thus, $(n/3 -1)^1 = 2n/3 -1$.

Otherwise, suppose that n/3 +1 is divisible by 3, then (n/3 -1)(n/3 -1) = (n/3)(n/3 +1) -n +1= n (n/3 +1)/3 -n +1 = n(0) -0 +1 (mod n) =1. That is, $(n \cdot 3 -1)^1 = n/3 -1 \pmod n$. □

Lemma 5.3: Let n be divisible by 3, and let n/3 +1 be a coprime to n. Then, the inverse of n/3 +1 (mod n) is equal to either (2n/3 +1) or (n/3 +1) depending on whether or not n is divisible by 9; respectively.

Proof: The proof methodology for this lemma is similar to that given in Lemma 5.2, and therefore omitted. □

Theorem 5.1: For any k ≥ 1, we can form a k-ft solution H of the distributed loop G(1, n-2: n) as follows.

a) If 3 divides (n+1), the solution H will be equal to H((n-2)/3, (n+1)/3, (n+1)/3 +1,.., (n+1)/3 +k : n+k).
b) If 3 divides (n-1), the solution H will be equal to H((2n-2)/3, (2n+1)/3, (2n+1)/3 +1 ,.., (2n+1)/3 + k : n+k).
c) If 3 divides n and gcd(n, (n/3 +1)(n/3 -1))=1, H will be equal to H((2n/3)-1, 2n/3,.., (2n/3)+k-1, (2n/3)+2, (2n/3)+3,.., (2n/3)+2+k : n+k).
d) If 3 divides n and gcd(n, n/3 +1) ≥ 1, H will be H((n/3) -1, (n/3),.., (n/3) -1+k, (n/3) +2, (n/3) +3,.., (n/3) +2+k : n+k).
e) If 3 divides n and gcd(n, n/3 -1) ≥ 1, H will be H((n/3) -2, (n/3) -1,.., (n/3) -2+k, (n/3)+1, (n/3)+2,.., (n/3)+1+k : n+k).

(Thus, in all cases, H has either k+2 or k+4 jumps.)

Proof: Suppose first that Case(a) is true, i.e., let (n+1) be divisible by 3. Then, $3^1 \pmod n$ will be equal to (n+1)/3, i.e., we can group the two jumps of G into a 3-distance partition, and then form a block graph Q for G of the form $Q(1, 3^1, (n-2), 3^1 : n) = Q(1, (n+1)/3, (n-2), (n+1)/3 : n) = Q((n-2)/3, (n+1)/3 : n)$, (where the multiplication is done mod n). Thus, a k-ft H of G will be H((n-2)/3, (n+1)/3, (n+1)/3 +1,.., (n+1)/3 +k : n+k).

Similarly, suppose that Case(b) is true, i.e., let (n-1) be divisible by 3, then it is not difficult to verify that $3^1 = (2n+1)/3$. This is because, $3 \cdot (2n+1)/3 = 3 \cdot (2n-2+3)/3 = 3 \cdot 2(n-1)/3 +3 = 2n-2+3 = 1 \pmod n$. Thus, we can group the jumps of G

into a 3-distance partition, and then form a block graph Q for G of the form Q(1..3[1], (n-2)..3[1]: n) = Q(1..(2n+1)/3, (n-2)..(2n+1)/3: n) = Q((2n-2)/3, (2n+1)/3: n), (where the multiplication is done mod n). Accordingly, a k-ft solution H of G is equal to H((2n-2)/3, (2n+1)/3, (2n+1)/3 +1,.., (2n+1)/3 +k : n+k).

Suppose Case(c) is true, i.e., let 3 divides n and gcd(n, (n/3 +1)(n/3 -1)) =1. Then, we can group the jumps of G into either an (n/3 +1)-distance partition or (n/3 -1)-distance partition. Both lead to solutions of equal cost, and therefore, we chose the latter, that is, the block graph of G in this case will be Q(1..(n/3 -1)[1], (n-2)..(n/3 -1)[1]: n). Since neither (n/3 +1) or (n/3 -1) is divisible by 3 in this case, therefore, n must be divisible by 9, i.e., by Lemma 5.2, the inverse of (n/3 -1) will be equal to (2n/3 -1). Thus, the block graph Q will be Q(1..(2n/3 -1), (n-2)..(2n/3 -1) : n) = Q((2n/3 -1), (2n/3 +2) : n) (where multiplication is done mod n). Thus, a k-ft H of G will be H((2n/3)-1, 2n/3,.., (2n/3)+k-1, (2n/3)+2, (2n/3)+3,.., (2n/3)+2+k : n+k).

Similarly, suppose that Case(d) is true, i.e., let 3 divides n and gcd(n, n/3 +1) ≠ 1. Then, by Lemma 5.1, n and (n/3)-1 must be coprime, i.e., we can group the jump of G into an (n/3 -1)-distance partition, and then form a block graph Q for G of the form Q(1..(n/3 -1)[1], (n-2)..(n/3 -1)[1] : n), where the inverse of (n/3 -1) in this case will be equal to (n/3 -1) as was shown in Lemma 5.2. Thus, the block graph is equal to Q(1..(n/3 -1), (n-2)..(n/3 -1) : n) = Q((n/3 -1), (n/3 +2) : n) (where multiplication is done mod n). Thus, a k-ft H of G will be H((n/3)-1, (n/3),.., (n/3)-1+k, (n/3)+2, (n/3)+3,.., (n/3)+2+k : n+k).

Finally, suppose that Case(e) is true, i.e., let 3 divides n and gcd(n, n/3 -1) ≠ 1. Then, by Lemma 5.1, n and (n/3 +1) must be coprime, i.e., we can group the jumps of G into an (n/3 +1)-distance partition, and form a block graph for G of the form Q(1..(n/3 +1)[1], (n-2)..(n/3 +1)[1] : n), where the inverse of (n/3 +1) in this case is equal to (n/3 +1) as was shown in Lemma 5.2. Thus, the block graph will be equal to Q(1..(n/3 +1), (n-2)..(n/3 +1) : n) = Q((n/3 +1), (n/3 -2) : n) (mod n). Therefore, a k-ft H of G will be H((n/3)-2, (n/3)-1,.., (n/3)-2+k, (n/3)+1, (n/3)+2,.., (n/3)+1+k : n+k). □

Table 5.1 For each case (a) to (e) proven in the above theorem, an example of a graph and its k-ft is given.

Graph G(1,n-2:n)	A 6-ft H of G
G(1, 9 : 11)	H(3,4,..,10: 17)
G(1, 14 : 16)	H(10,11,..,17: 22)
G(1, 16 : 18)	H(11,12,..,20: 24)
G(1, 13 : 15)	H(4,..,13: 21)
G(1, 10 : 12)	H(2,3,..,11: 18)

Actually, the solutions developed in the above theorem are either optimal (when n is not divisible by 3) or nearly-optimal (when 3 divides n). The proof follows from the following theorem which establishes a lower- bound on the node-degree of any k-ft solution. (Notice that the node degree of a loop H is twice the number of its jumps.)

Theorem 5.2: Given a distributed loop G with d jumps, any k-ft H of G must have a degree ≥ 2(d+k).
Proof: Suppose to the contrary a k-ft solution H has a smaller degree than 2(d+k). Then, for any node u, either its in-degree(u) or out-degree(u) is less than (d+k). Suppose, for example, in-degree(u) < (d+k) and select any set of k nodes "adjacent to u" as being faulty. Excluding or removing these k nodes from H will make the new value of in-degree(u) less than d, i.e., the graph obtained by excluding these k nodes from H cannot contain a subgraph isomorphic to G, which implies that H cannot be a k-ft of G leading to a contradiction. □

As a consequence of the lower-bound proven above, the k-ft solution constructed by our Theorem 2.1 for a loop G with a single jump (or with a single block of jumps) is optimal in node-degree. That is, G(b :n) has an optimal k-ft of the form H(b, b1,.., bk : nk).

6. CONCLUSIONS

Distributed loop networks form the underlining topology of many local area networks and some parallel computers (such as the ILLIAC machine). This class of networks satisfies many useful properties and includes as special cases) important topologies such as rings and circulant graphs, for examples.

Due to the highly regular structure of distributed loop networks, they tend to be vulnerable to node failures. Accordingly, we have examined the problem of extending these networks by adding spare nodes and links, so as to make the structure fault-tolerant. Our method can be used to tolerate any desired

number of node failures, and moreover, it can be applied whether the parameters that define the distributed loop are stated numerically (as constants) or symbolically (as variables).

REFERENCES

[1] Anderson, T., and P. Lee, "Fault-Tolerance Principles and Practice", Prentice-Hall International, London, 1981.

[2] J. Bermond, F. Comellas and D. Hsu, "Distributed Loop Computer Networks A Survey", J. of Parallel and Dist. Computing, 24, 1995, pp. 2-10.

[3] Bruck, J., R. Cypher and C. Ho, "Fault Tolerant Meshes and Hypercubes With Minimal Number of Spares", IEEE Trans on Comp, 42, no. 9, September 1992, pp. 1089-1103.

[4] Chuang, Y., L. Hsu and C. Chang, "Optimal 1 edge Fault-Tolerant Designs for Ladders", Info. Proc. Letters, 84, no. 2, October 2002, pp. 87-92.

[5] Chung, F., F. Leighton, and A. Rosenberg, "Diogenes: a Methodology for Designing Fault Tolerant VLSI Processor Arrays", in Proc. IEEE 13th Conf on Fault Tolerant Computing Symp, June 1993, Chicago, IL, pp. 26-32.

[6] Dutt, S., and J. Hayes, "Designing Fault-Tolerant Systems Using Automorphisms", J. on Parallel and Dist Computing, 12, no. 3, July 1991, pp. 249-268.

[7] Farrag, A., "Algorithm for Constructing Fault-Tolerant Solutions of the Circulant Graph Configuration", in Proc. of 5th IEEE Symp. On Frontiers of Massively Parallel Computations, February 1995, McLean, Virginia, pp. 514-520.

[8] Farrag, A., and R. Dawson, "The Fault Tolerant Extension Problem for Complete Multipartite Networks," in IEEE Trans. on Parallel and Dist. Sys., 5, no. 2, February 1994, pp. 205-210.

[9] Farrag, A., and S. Lou, "Applying Fault Tolerant Solutions of Circulant Graphs to Multi Dimensional Meshes" in Computers & Mathematics Journal, 50, no. 8-9, November 2005, pp. 1383-1394.

[10] Farrag, A., S. Lou and Y. Qi, "Fault tolerance and Reconfiguration of Circulant Graphs and Hypercubes", *in Proc. Of High Performance Computing Symposium (HPCS-2008)*, Ottawa, Ontario, Canada, Apr 2008, pp. 475-481.

[11] Grnanov, A., L. Kleinrock, and M. Gerla, "A highly Reliable Distributed Loop Network Architecture", in Proc. IEEE Symp. Fault Tolerant Computing, Kyoto, Japan, October 1980, pp. 319- 324.

[12] Lipson, J., "Elements of Algebra and Algebraic Computing", Benjamin/Cummings Publisher, Melno Park, California, 1981.

[13] Liu, M. T.,"distributed Loop Computer Networks", Advances in Computers, Vol. 17, Press, New York, 1978, pp. 163-221.

[14] Peha, J., and F. Toubagi, "Analyzing the Fault Tolerance of Double-Loop Networks", IEEE/ACM Trans. On Networking, V. 2, N. 4, August 1994, pp. 363-373.

[15] Raghavendra, C., M. Gerla and A. Avizienis, "A Reliable Loop topologies for Large Local Computer Networks", *IEEE Trans. Computer*, V. 34, N. 1, Jan 1985, pp. 46-55.

[16] Raghavendra, C., and J. Silvester, "Double Loop Network Architectures- A Performance Study", *IEEE Trans. Comm.*, V. 33, N. 2, Feb 1985, pp. 185-187.

[17] Rennels, D., "On Implementing Fault Tolerance in Binary Hypercubes", Digest of papers of IEEE Symp on Fault-Tolerant Comp, (July), Vienna, Austria, 344-349.

[18] Schmitter, E., and P. Baues, "The Basic Fault Tolerant System", IEEE Micro, 4, no. 1, February 1984, pp. 66-74.

[19] Snyder, L.,"Introduction to the Configurable, Highly Parallel Computer," IEEE Computer, 15, no. 1, January 1992, pp. 47-56.

[20] Sung, T., M. Lin, and T. Ho, "Multiple-Edge Fault Tolerance with respect to Hypercubes", IEEE Trans Parallel and Dist Systems, 8, no. 2, February 1997, pp. 187-192.