Fault-Tolerant Routing for Pyramid Networks Using
Least Level Minimal Routing Method

Da-Ren Chen
Department of Management Information System, Hwa Hsia College of Technology and Commerce, Taipei, Taiwan, R.O.C.
No.111, Huashin St., Junghe City, Taipei 235, Taiwan
danny063@ms13.hinet.net

Chiun-Chieh Hsu
Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.
43 Section 4, Kee-Lung Road, Taipei 106, Taiwan
cchsu@cs.ntust.edu.tw

Abstract

Pyramid networks have long been proposed for parallel processing. However, when the network contains multiple pairs of source and destination nodes, it is often difficult to use the pyramid efficiently since most of the algorithms try to simultaneously use the apex for each pair of nodes. Therefore, there may exist a severe bottleneck in such circumstances. We propose a Least Level Minimal Routing (LLMR) scheme for releasing the bottleneck in pyramid networks. At each vertex, the algorithm using LLMR needs only $O(n)$ time to determine the shortest path from the source to the destination, where $n$ denotes the number of levels in the nonfaulty pyramid network. Besides, LLMR releases the apex from almost all transmission load for any pair of nodes. Furthermore, it reduces enormously the traffic in higher levels of the pyramid. We also provide point-to-point node disjoint parallel routing algorithm in pyramid networks, where the lengths of the paths are at most $4n-2$ steps.

Keywords
pyramid networks, fault tolerant, minimal routing, load balancing.

1 Introduction

The pyramid network is the conjunction of meshes and tree structures. The essential nature of a pyramid of size $e$ has an $e^{1/2} \times e^{1/2}$ mesh-connected computer as its base, and $\log_e(e)$ levels of mesh-connected computers above. A processing element (PE) at level $k$ is connected to 4 siblings at level $k$ and 4 children at level $k+1$ and a parent at level $k-1$, where $k \in N$ (see Figure 1).

The pyramid network is one of the important structures in parallel and network computing [7], because some parallel algorithms can be efficiently implemented on pyramid networks. There is no reason to confine pyramid computers to low-level image processing. They can be adapted to many other problems and ought to be considered as substitutes for machines such as the mesh-connected computer [11,12,13]. For example, Russ et al. [5,6], and Quentin [10] proposed several pyramid computer algorithms which are significantly faster than their mesh-connected computer counterparts. These algorithms solve problems in graph theory, image processing, and digital geometry. In addition, Feng Cao et al. [2] studied the fault-tolerant properties of pyramid networks quantitatively in terms of the new concepts such as fault diameter and w-wide diameter [4], and proposed an algorithm for fault-tolerant routing in pyramid networks.

Routing algorithms for the Pyramid networks are described in [2,5]. Feng Cao et al. [2] has shown constructively that the connectivity and fault-tolerant properties of pyramid networks. Their useful proof directly leads to an end-to-end fault-tolerant algorithm between all pairs of distinct nodes. They presented an algorithm with $O(n^2)$ time complexity for building containers between two nodes in PM[$n$] and minimize the lengths of the containers, where $n$ is the cardinality of levels in the pyramid network PM. In fact, it is possible to find out a more efficient routing algorithm by way of limiting the lengths of those parallel paths, which can be very close to the fault diameter. Since pyramids do not have very good load balancing characteristic [19], a severe bottleneck may happen in the apex of a pyramid when the network contains multiple pairs of source and destination nodes between which messages are transmitted along their...
minimal path. In order to obtain a routing algorithm with less time complexity and better traffic condition in the apex and higher levels of the pyramids, it is necessary to propose different approaches.

This paper is devoted particularly to a kind of routing scheme, the so called Least-Level-Minimal-Routing (abbreviated to LLMR). LLMR guarantees that the minimal path between any pair of source and destination nodes will adopt the path in the lowest level of the pyramid among all minimum paths between them. Furthermore, in order to take advantage of LLMR, we build an efficient fault-tolerant routing algorithm, where three node-disjoint paths can be constructed between any pair of nodes in the faulty pyramid networks. The time complexity of the proposed algorithms are $O(n)$.

Our algorithm need not compute the whole shortest path in advance. Instead it dynamically computes the next vertex in the shortest path that the message is to be sent. At each vertex, for any destination, our algorithm needs only constant time and space to determine the next vertex on the shortest path to which to message must be sent. Therefore, our approach can be more adaptive to both edge-fault and vertex-fault.

The rest of the paper is organized as follows: In Section 2, we give some notations and definitions to pyramid networks and LLMR scheme. Some important theorems about LLMR and an end-to-end minimal routing algorithm are presented in Section 3. Section 4 offers the fault-tolerant routing algorithm based on the LLMR scheme. Finally, we conclude this paper in Section 5.

2. Notations and definitions

Most of the notations and definitions of a pyramid will follow those in [2]. A mesh $M[a,b]$ is composed of a set of nodes, $V(M[a,b])=\{(x,y)|1 \leq y \leq b, 1 \leq x \leq a\}$, and there is an edge between two nodes $(x,y)$ and $(x',y')$ iff $|x-x'|+|y-y'|=1$. A pyramid network of $n$ levels, denoted by $PM[n]$, is a set of nodes $V(PM[n])=\{(k;x,y)|0 \leq k \leq n, 1 \leq x,y \leq 2^k\}$. A node $(k;x,y) \in V(PM[n])$ is said to be a node at level $k$. All the nodes in level $k$ constitutes a mesh $M[2^k,2^k]$. Node $(k;x,y) \in V(PM[n])$ is also connected to $(k+1;2x-1,2y-1),(k+1;2x,2y-1),(k+1;2x-1,2y-1)$ and $(k+1;2x,2y)$ for $0 \leq k < n$. For ease of reference, some notations are summarized in the following: $P^j(k;x,y)$ denotes that the node $(k;j;x',y')$ is said to be $P^j(k;x,y)$ if there exists $j$ uplinks continuously from node $(k;x,y)$ to node $(k;j;x',y')$.

An important characteristic of the nodes in pyramid networks is that the relation between node $s=(k;x,y)$ and its parent is $P^j(s)=(k-1;x',2y')$. For general and clear presentation, we can modify it as $P^j(s)=(k-j;x,2y')$ where $0 \leq j \leq k$, where $s$ is the $j$th generational descendant of node $P^j(s)$. For example, in Figure 1, given a node $s=(2;3,1)$ in $PM[2]$, we can exploit the equation above to know the grand parent of node $s$ is $P^2(s)=(2;2,3)$, i.e. the apex of the pyramid. Moreover, we have following notations to describe every node in a pyramid. $l(t)$ denotes the level to which node $t$ belongs. $dist(s,d)$ is the shortest distance between nodes $s$ and $d$ in the same level. $CS(t)$ : Covering Square is a set consisting of all nodes in the most bottom level $n$ belonging to the descendants of node $t$, where $l(t) < n$. If $s$ and $d$ are distinct nodes in the same level then $CS(s)$ and $CS(d)$ are disjoint sets. $cs_{max\_dist}(s,d)$ is an integer representing the maximum distance between nodes $s$ and $d$ which belong to $CS(s)$ and $CS(d)$, respectively. $cs_{min\_dist}(s,d)$ is an integer representing the minimum distance between nodes $s$ and $d$ which belong to $CS(s)$ and $CS(d)$, respectively. $diam(u,v)$ presents the diameter of the subgraph ranging from level $u$ to level $v$ ($n \geq u \geq v$) in a pyramid.

Example. Figure 2 shows a $PM[2]$. In Figure 2, let $s$ and $d$ be $(1;1,2)$ and $(2,1,2)$ respectively. It can be easily seen $dist(s,d)=2$. Note that the $dist(s,d)$ exists only if $l(d)=l(s)$. $CS(s)$ contains nodes $(2;1,3)$, $(2;1,4)$, $(2;2,3)$ and $(2;2,4)$. $CS(d)$ contains nodes $(2;3,1)$, $(2;4,1)$, $(2;3,2)$ and $(2;4,2)$. The results of $cs_{min\_dist}(s,d)$ and $cs_{max\_dist}(s,d)$ are two and six, respectively. Besides, assuming that $w=0$, $w=1$ and $w=2$, we can obtain $diam(u,v)=2$, $diam(w,u)=4$, $diam(u,w)=6$ and so on. At the final notation, assuming that $s'$ and $d'$ are source and destination nodes respectively, and the $i$-th path between $s'$ and $d'$ contains nodes $(2;1,4)$, $(1;1,2)$, $(0;1,1)$, $(1;2,1)$ and $(2;4,1)$, then $path\_height(s',d')=\{2,1,0,1,2\}$.

Definition 1: Given a pair of source node $s$ and destination node $d$ in $PM[n]$. The highest level of a minimum path routed by LLMR algorithm between
nodes \( s \) and \( d \) will not be higher than level of
\[
limited_height = \max\{\min\{\text{path_height}(s,d)\}\}\]

Example. See Figure 2 for the detailed description of above.
The locations of source node \( d' \) and destination node \( s' \) are \((2;4,1)\) and \((2;2,3)\), respectively. A shortest path between \( d' \) and \( s' \) constructed by general algorithms may belong to one of the following groupings: \((2;4,1)\rightarrow(1;2,1)\rightarrow(0;1,1)\rightarrow(1;1,2)\rightarrow(2;2,3)\) or \((2;4,1)\rightarrow(1;2,1)\rightarrow(1;1,1)\rightarrow(1;1,2)\rightarrow(2;2,3)\), etc. In other words, it is possible for the algorithms to route the messages through the minimal path among level 0 and level 2. For the LLMR algorithm, it has one of the following combinations: \((2;4,1)\rightarrow(2;3,1)\rightarrow(2,2,1)\rightarrow(2,2,2)\rightarrow(2;2,3)\) or \((2,4,1)\rightarrow(2,4,2)\rightarrow(2;4,3)\rightarrow(2,3,3)\rightarrow(2;2,3)\), etc. That is, in addition to satisfying the condition of minimizing the length of a path, LLMR will limit the highest level which the message should go through.

### 3. The LLMR approach

For the sake of satisfying the conditions of LLMR, the most important thing is to efficiently find a feasible level of a pyramid when we try to find a minimum path between two nodes in \( PM[n] \). In this section, we first give some theorems about the distance relations between source and destination nodes. It helps us to easily find a proper level in a pyramid network. In order to employ LLMR, we give a routing algorithm to find a path between any two nodes in \( PM[n] \) when there are no faulty nodes in it. The algorithm needs only \( O(n) \) time to construct an minimal path between any two nodes.

The underlying target architecture is assumed to be homogeneous, and the communication bandwidth between any pair of PEs is considered to be the same. In addition, it is assumed that there is no fault in the system.

#### 3.1. The Properties of \( cs\_min\_dist(s,d) \) and \( cs\_max\_dist(s,d) \)

First, we establish two theorems which describe the distance relations between two covering squares \( CS(s) \) and \( CS(d) \).

**Theorem 1.** For any pair of distinct nodes \( s=[u;x,y] \) and \( d=[u;x,y] \) in the same level, \( u<\hat{n}, i=dist(s,d) \) and \( j=\hat{n}-u \) we have
\[
\text{cs\_max\_dist}(s,d) = (2^i-1)*((i+2)+i)
\]

**Proof:** Obviously, it is true for \( i=0 \) and \( j=1 \). That is, \( cs\_max\_dist(s,d) \) is the diameter of a 2x2 mesh. It can be easily verified that, for \( i=dist(s,d)=1 \) and \( j=\hat{n}-u=1 \), \((2^1-1)(14+2)+1=4\).

Suppose it is true for \( n-u=j-1 \) and \( i=1 \). By the properties of a pyramid, there must exist two nodes \( \alpha \) and \( \beta \) belonging to \( CS(s) \) and \( CS(d) \) respectively and \( dist(\alpha,\beta) = (2^i-1)x3+1 \). Without loss of generality, we assume that the positions of \( \alpha \) and \( \beta \) are \((n;1,1)\) and \((n;2^i-1)x3+1+w, w+1 \), where \( 0\leq w \leq (2^i-1)x3+1 \). The sons of \( \alpha \) are \((n+1;1,1)\), \((n+1;2,1)\) and \((n+1;2;2)\) and those of \( \beta \) are \((n+1;2^i-1)x3+1+w, w+1 \), \((n+1;2^i-1)x3+1+w, w+1 \) and \((n+1;2^i-1)x3+1+w, w+1 \). We select a pair of nodes \( \alpha=\alpha=(n+1;1,1) \) and \( \beta=\beta=(n+1;2^i-1)x3+1+w, w+1 \) above which, have the longest distance
\[
\text{dist}(\alpha',\beta')=(2((2^i-1)x3+1+w)+2(w+1)-1)
\]
\[
=6(2^i-1)+4
=3(2^i-1)+4
=3(2^i-1)+1=(2^i-1)(1+2)^{i-1} \ldots \ldots \ldots (1)
\]
This shows that \( j \) follows from \( j-1 \) when \( i=1 \). By the use of equation (1), it is trivially true for \( j \) and \( i=1 \). Note that we still have another variable \( i \). Suppose it is true for \( i-1 \) and \( j \); that is, there are two nodes \( \alpha \) and \( \beta \) of which their distance is \((2^i-1)x(i+2)+2(i-1)\) and \( \alpha \) and \( \beta \) belong to \( CS(s) \) and \( CS(d) \) respectively. To show that \( dist(\alpha',\beta')=i \) is true, we add one link and the width of one covering square to \( dist(\alpha,\beta) \). Then we obtain
\[
\text{dist}(\alpha,\beta)=(2((2^i-1)x3+1+w)+2(w+1)-1)
\]
\[
=(2^i-1)x(i+2)+2(i-1)+2
\]
\[
=(2^i-1)x(i+2)+i \]
This establishes the inductive step of the proof. Thus, the equality for the \( cs\_max\_dist(s,d) \) is valid for \( 0\leq i \) and \( 0\leq j \).

**Theorem 2.** For any pair of distinct nodes \( s=[u;x,x]\) and \( d=[u;x,y] \), \( u<n, i=dist(s,d) \) and \( j=n-u \) we have
\[
\text{cs\_min\_dist}(s,d) = (2^i-1)+(i+2)+i
\]
\[
=2^i-1+(i+2)+i \]
\[
=2^i-1+(i+2)+i \]

**Proof:** We prove equation (2) only, whereas equation (3) can be proved in a similar way. If \( i=1 \) and \( j=1 \), it is easy to see that \( cs\_min\_dist(s,d)=1 \) is true. Such two covering squares are tiled by placing them on the adjacent locations. Assume the induction hypothesis that the theorem is true for \( i-1 \) and \( n-u=j-1 \). Obviously, in order to take advantage of the regular structure of the pyramids and the definition of covering square in section 2, \( CS(s) \) and \( CS(d) \) are adjacent covering squares. There must exist two nodes \( \alpha \) and \( \beta \) belonging to \( CS(s) \) and \( CS(d) \) and the distance between them is one. Since \( \alpha \) and \( \beta \) are neighboring nodes, the distance between their closest sons is one as well. Because the theorem is true for \( i=1 \) and \( j=1 \), and for \( j>0 \) and \( i=1 \), it is true for \( i=1 \) and \( j=1 \). By applying the result above, we suppose the induction hypothesis that the theorem is true for \( i-1 \) and \( j \). Suppose the distance between two nodes \( \alpha \) and \( \beta \) is \((2^i-1)+(i+2)+i \) and they belong to \( CS(s) \) and \( CS(d) \) respectively. Similarly, we add one link and the width of one covering square to \( dist(\alpha,\beta) \) to obtain
\[
\text{dist}(\alpha,\beta)=(2^i-1)+(i+2)+1 \]
The following property can be derived directly from fact, for any pair of distinct nodes increase the diameter of a pyramid or subpyramid. In

diam
e
We prove this theorem by induction on \(e\) greater than two.

The following lemmas organize the properties of

Furthermore, we provide Theorem 6 that presents a constant bound of maximum value of \(i\).

**Lemma 3:** Let there exist two distinct nodes \(s=(u,x_i,y_i)\) and \(d=(u,x_j,y_j)\) and \(u\neq s, n, j=n-u, i \geq 2\). If \(x_i \neq x_j\) and \(y_i \neq y_j\), then \(cs_{max\_dist}(s,d)\) and \(cs_{min\_dist}(s,d)\) are positive even numbers.

**Proof:** Let \(j=1\) and \(i \geq 2\), by the use of theorem 1, we have

\[cs_{max\_dist}(s,d) = (2^{i-1}) \times (i+2) + i = 2i + 2\]

and

\[cs_{min\_dist}(s,d) = (2^{i-1}) \times (i+2) + i = 2i - 2\]

When \(x_i \neq x_j\) and \(y_i \neq y_j\), hence, \(cs_{max\_dist}(s,d)\) and \(cs_{min\_dist}(s,d)\) are positive even numbers as long as the value of \(i\) is greater than two.

To make use of Theorem 1, the following lemma can be easily proved.

**Lemma 4:** For \(e \in N, e \geq n\), \(diam(e, 1) = diam(e, 0)\)

**Proof:** We prove this theorem by induction on \(e\). Let \(a\) and \(b\) be distinct nodes in level 1 of a pyramid, i.e. level \(e\), then \(P^1(a) = P^1(b)\), \(dist(P^1(a), P^1(b)) = 0\). Since \(diam(1, 0) = 2\), \(diam(1, 0) = diam(1, 1) = 2\). Assume that it is true for \(e-1\), so that \(diam(e-1,1) = diam(e-1,0) = 2\) as \(e=1\). Moreover, \(diam(e,1) = diam(e-1,1) + 2 = 2 + 2e = diam(e,0)\).

Lemma 4 states that deleting the apex will not increase the diameter of a pyramid or subpyramid. In fact, for any pair of distinct nodes \(\alpha = (u,x,y)\) and \(\beta = (y,x,y)\), the minimal paths between \(\alpha\) and \(\beta\) need not pass through the apex of a subpyramid, for \(0 < u, v \leq n\). The following property can be derived directly from the definition of pyramid networks.

**Property 5:** Let the source node \(s\) and destination node \(d\) be in level \(k\). If \(dist(s,d) \geq 3\) then the length of any path which goes through the level \(k-j\) \((0 \leq j \leq k)\) between \(s\) and \(d\) will not be shorter than \(3\).

That is, according to the definition of LLMR, for any pair of nodes \(s\) and \(d\) in level \(k\) whose \(dist(s,d) \geq 3\), LLMR algorithm chooses the paths in level \(k\).

**Theorem 6:** Let \(k \in N\), \(k > 0\) and the corresponding parents of two distinct nodes \(s\) and \(d\) in level \(k\) be \(P^1(s) = (k-1; x^s, y^s)\) and \(P^1(d) = (k-1; x^d, y^d)\). If \(dist(s,d) = u \geq 6\), then \(dist(P^1(s), P^1(d)) + 2 < u\).

**Proof:** In Figure 4, let \(r, u \in N\). We treat level \(k\) as a base of a subpyramid. Therefore, according to Theorem 1 and Theorem 2, let \(j=1\) and \(cs_{min\_dist}(P^1(s), P^1(d)) \geq 7\). We can calculate \(i, i = dist(P^1(s), P^1(d))\). There are two possible cases here:

**Case 1**: When \(x^s = x^d(d)\) and \(y^s = y^d(d)\), provided that \(r=1\), we have \(cs_{min\_dist}(P^1(s), P^1(d)) = (2^{i-1}) \times (i-1) = (i-1) + i \leq (6+r)\) and, \(i=3\). Then \(i+2 = 5 \leq 7\). This is true for \(r=1\). Assume that \(r\) holds, it follows that \(2^{i-1}) \times (i-1) + i \leq 6 + r\) and \(i+2 < 6 + r\).

\[i \leq (7+r)/2\]

This shows that \(r+1\) follows from \(r\).

**Case 2**: When \(x^s \neq x^d(d)\) and \(y^s \neq y^d(d)\),

From the proof of Lemma 3, the value of \(cs_{min\_dist}(s,d)\) is an even positive number when \(j=1\) and \(i \geq 2\). We need to consider the following two cases:

**Subcase 2.1:** \(u = dist(s,d)\) is an even number. Provided that \(r=1\) and \(u = 6 + 2r\), we have \(cs_{min\_dist}(P^1(s), P^1(d)) = (2^{i-1}) \times (i-1) + i \leq (6+2r)\) and \(i=5\). Then \(i+2 = 7 \leq u=8\). This is true for \(r=1\). Assume that \(r\) holds, so that \(cs_{min\_dist}(P^1(s), P^1(d)) \leq 6 + 2r\), \(i=2 + 6 + 2r\). It must be shown that \(r+1\), which states that \(i+2 < 6 + 2(r+1)\) must also be true under this assumption. This can be done since

\[cs_{min\_dist}(P^1(s), P^1(d)) \leq 6 + 2r + i + 2 < 6 + 2(r+1)\]

When \(r=r+1\), \((4 + (r+1)) \times 2 \leq 6 + 2(r+1)\). This inequality is still valid.

**Subcase 2.2:** \(u = dist(s,d)\) is an odd number. Assume that \(r=1\) and \(u = 6 + 2r\), we have \(cs_{min\_dist}(P^1(s), P^1(d)) = (2^{i-1}) \times (i-2) + i \leq (6+2r+1)\) and \(i=4\). Thus, \(i+2 = 6 < u=7\). It is true for \(r=1\). Assume that it is true for any pair of nodes \(s\) and \(d\) with \(dist(s,d) = u = 6 + 2r\). We show that it is also true for \(u = 6 + 2(r+1)\). This can be done since

\[cs_{min\_dist}(P^1(s), P^1(d)) \leq 6 + 2r + i + 2 < 6 + 2(r+1)\]

When \(r=r+1\), \((3 + (r+1)) \times 2 \leq 6 + 2(r+1)\). This inequality is still valid. Thus, if \(dist(s,d) = u \geq 6\), then \(dist(P^1(s), P^1(d)) + 2 < u\).
Among the formulation $dist(P'(s),P'(d))+2< u$, ‘2’ represents one uplink and one downlink which are part of the minimal paths using level k-1.

In Figure 4, since $cs_{max} \cdot dist(P'(s),P'(d)) \geq cs_{min} \cdot dist(P'(s),P'(d))$ when $i, j \geq 0$ and $i, j \in \mathbb{N}$, we need only to show that $i+2< dist(s,d)$ when $dist(s,d) \geq 7$, where $i$ is derived from $cs_{min} \cdot dist(P'(s),P'(d))$. For example, given a pair of nodes $s$ and $d$ in level $k$ which belong to $CS(P'(s))$ and $CS(P'(d))$ respectively and $dist(s,d)$ is equal to the minimum distance between $CS(P'(s))$ and $CS(P'(d))$, i.e. no any other pairs of nodes within $CS(P'(s))$ and $CS(P'(d))$ can get closer than $dist(s,d)$.

The above theorem is particularly important. Let $s$ and $d$ be distinct nodes in level $k$. It reveals that there possibly exists a minimal path passing through level $k$-$j$, provided that $dist(P'(s),P'(d)) \leq 6$. A path getting across level $k$-$j$ is not a minimum path if $dist(P'(s),P'(d)) \geq 6$, because one path passing through level $k$-$j$-1 will certainly be shorter than it. Therefore, according Theorems 4 and 6, $dist(P'(s),P'(d))$ should be constrained within the constant tight bounds between one and six. That is, make use of the equation in Theorem 1, we have

$$cs_{max} \cdot dist(s,d) = (2-1) \times (i+2) + i \geq dist(s,d);$$

For every $i$, there is a corresponding value $j$ which indicates that one message starting from source node $s$ and using $i$ uplinks to pass through level $k$-$j$ to destination node $d$ may results in a minimal path. Now note that the $j$ values are only for reference. These values are meaningless and actually are not present. For example, $dist(s,d)=11$ in Figure 5, and making use of the above iterations, we have six pairs of $[i, j]: [1,3], [2,2], [3,2], [4,2], [5,1]$ and [6,1]. But for $i=5, j=2$, it is not possible that $dist(s, P'(s), P'(d))=3$. For the sake of examining the existence of pairs $[i, j]$, we should test every pairs of $[i, j]$ by using the equation $P'(T)=\min(j \times 2^2, [y_2^2], \text{where } T \text{ is a source node or destination node in level n. If the value } i \text{ equals to its corresponding } dist(n-j, x_2^2), [y_2^2], (n-j) \times 2^2, [y_2^2]);$ we can conclude that the pair $[i, j]$ is present and level $n-j$ is called feasible level.

Let nodes $s$ and $d$ be in the levels $p$ and $q$ in $PM[n]$ respectively. The theorem listed below gives a promise that the feasible level $k$ satisfies $k \leq \min\{p, q, n\}.

**Theorem 7 :** Let $s$ and $d$ be distinct nodes in level $p$ and $q$ respectively and $p < q < n$. The minimal path between $s$ and $d$ always go through the level $k$ for $k \leq p$.

### 3.2. An Minimal Routing Algorithm

In the following, we give an algorithm for constructing a minimal path between any two nodes in $PM[n]$ using LLMR method.

**LLMR algorithm**

1. set Source $s=(k, s_r, s_i)$, Destination $d=(m, d_r, d_i)$
2. If $k < m$
   1. $d^{'1} = d^{1}/_2$, $d^{'2} = d^{2}/_2$, $p^{opt}(d) = \text{opt}_{LLMR}(k, s, p^{opt}(d))$
   2. use LLMR(uplink) towards $P^{opt}(d)$
   3. use X-Y routing or Y-X routing to $P^{opt}(d)$
   4. use $m(k-opt)$ downlinks towards $d$
3. If $m < k$
   1. $s^{1} = s^{1}/_2$, $s^{2} = s^{2}/_2$, $P^{opt}(s) = \text{opt}_{LLMR}(m, P^{opt}(d))$
   2. use $m$-opt uplinks towards $P^{opt}(s)$
   3. use X-Y routing or Y-X routing to $P^{opt}(d)$
   4. use $d(k-opt)$ downlinks towards $d$
4. If $k = m$
   1. $opt_{LLMR}(k, s, d)$
   2. use $X-Y$ routing or $Y-X$ routing to $P^{opt}(d)$
   3. use $d(k-opt)$ downlinks towards $d$
end of LLMR algorithm

**Procedure LLMR(k, s, d)**

1. if dist(s, d) $\leq 3$ then return 0
2. if dist(s, d) $\geq 4$
3. for $i=1$ to 6 do
   1. $(2^i-1) \times (i+2) + i \geq dist(s, d)$
   2. $P'(s) = \text{opt}(k-j, x_2^2, [y_2^2], P'(d))$
3. if $i \neq dist(s, d)$
   4. then feasible[i] $= \infty$
   5. else feasible[i]$ = i$
4. Min$= \text{feasible}[\min(2, \text{feasible}[i+1])$)
5. if Min$> 1$ then chose the smallest feasible[i]
6. return Min

In the following, we give an explanation for the algorithm given above. In step 1, for all pairs of distinct nodes $s=(k, s_r, s_i)$ and $d=(m, d_r, d_i)$ where $0<k, m \leq n$, we can separate them into three cases: $k < m$, $m < k$ and $m = k$. To simplify our presentation, we will refer to $\max(m, k)$ as $n$ of $PM[n]$. In step 2, as shown in Figure 4, we construct a path from $s$ to $d$ when $k < m$. First, we compute the location of $P^{opt}(d)$. Since nodes $s$ and $P^{opt}(d)$ are in the same level, we then treat $s$ and $P^{opt}(d)$ as a source $s$ and destination $d$ of Procedure LLMR(k, s, d). It then returns a value $opt$ that points out the level $k$-$opt$ will meet the conditions of LLMR. The way of finding the feasible level we mentioned before is in step 2 of procedure LLMR. After finding all the feasible levels in step 2.1, we can choose the minimal path in step 2.2 that goes across one of the feasible levels. In step 2.2 of LLMR algorithm, it uses $opt$ uplinks towards node $P^{opt}(s)$ (see (1) of Figure 4).

Next, it constructs a minimal path using X-Y routing (first $X$ and then $Y$) or Y-X routing [13] (first $Y$ and then $X$) between $P^{opt}(s)$ and $P^{opt}(d)$ in a mesh in level $k$-$opt$, see (2) of Figure 4). It is known that the length of the path between $P^{opt}(s)$ and $P^{opt}(d)$ is $i$.
and $0 < i \leq 6$. Finally, in step 2 (see (3) of Figure 4), in order to employ $m \cdot k + \text{opt}$ down -links, it can build a minimal path between $P^{m \cdot k + \text{opt}}(d)$ and $d$.

3.3. Running Trace of the Algorithm

This subsection illustrates the working of the procedure LLMR for the illustrated pyramid in Figure 5. In Figure 5, $s$ and $d$ are in the same level. Starting from step 4, the main program will call Procedure LLMR. Since $\text{dist}(s, d) = 11 > 6$, we have to calculate the corresponding value $j$ for every $i$, $0 < i \leq 6$ in step 3.1 of Procedure LLMR. Thus, all pairs of $(i, j)$ are $\{1,3\}, \{2,2\}, \{3,2\}, \{4,2\}, \{5,1\}$ and $\{6,1\}$. Steps 3.2 and 3.3 in Procedure LLMR are in charge of sifting the feasible levels from these pairs, where $\{2,2\}$ and $\{5,1\}$ are stored in array feasible[$i$]. Step 2.2 in Procedure LLMR finds a feasible level where $2 \cdot \text{feasible}[i] + i$ is minimal. In case that two minimal feasible levels exist, step 5 in Procedure LLMR will choose the level where $j$ is the smallest. Therefore Procedure LLMR returns $\text{opt} = 2$. After receiving $\text{opt} = 2$ in step 4.1, we then construct a path between $s$ and $P^{\text{opt}}(s)$ along nodes $(3,1,1), (2,1,1)$ and $(1,1,1)$. In step 4.2, one path between $P^{\text{opt}}(s) = (1,1,1)$ and $P^{\text{opt}}(d) = (1,2,2)$ may contain nodes $(1,1,1), (1,2,1), (1,2,2)$ or $(1,1,1), (1,1,2), (1,2,2)$. The former is called X-Y routing and the later is called Y-X routing. Finally, in step 4.3, the path uses only downlinks from $P^{\text{opt}}(d)$ to reach $d$. Thus, it contains $(1,2,2), (2,4,3)$ and $(3,8,5)$.

Theorem 8: LLMR constructs a minimal path between any two nodes in $PM[n]$ with $O(n)$ time complexity.

4. Construction of node-disjoint path

It is known that the line connectivity of a pyramid is three [2], where the line connectivity of a connected network is defined to be the maximum number of links whose removal does not affect the connectness of the network. In this section we give an algorithm to find three paths between two nodes in $PM[n]$ under the condition that there are at most two faulty nodes or two faulty communication links. The objective is to minimize the time complexity of building such three node-disjoint paths between any pair of nodes.

4.1. Available Levels for Node-Disjoint Paths

The following theorem provides a sufficient number of levels to make sure that each path can pass through these levels such that their distances are no more than six. According to the Procedure LLMR, we can figure out which level the minimal path and the second minimal path will pass.

Theorem 9: Let nodes $s$ and $d$ be distinct nodes in the level $k$ of $PM[n]$. If $\text{dist}(s, d) > 6$ then there exist $A = \{a \mid \text{dist}(P^a(s), P^a(d)) \leq 6 \}$ and $|A| > 2$.

Theorem 9 can be realized in Figure 6. There are at least three levels can be used to construct the three node-disjoint paths between any pair of distinct nodes $s$ and $d$ for $\text{dist}(s, d) > 6$. According to theorem 4, we will not consider the level $j$ in which $\text{dist}(P^j(s), P^j(d)) = 0$ for $\min(l(d), l(s)) > 1$. Therefore, in most cases, there are exactly two levels which can be employed by the algorithm to build the three node-disjoint paths between any pair of nodes. The following rule determine how the
two levels these three paths should be allocated to.

Figure 6. Feasible levels for node-disjoint paths in subpyramid

**Rule:** To use the algorithm LLMR, we obtain two levels in which one path exploits the higher level and the other two paths exploit another one.

For ease of exploiting Procedure LLMR in this section, the following procedure requires some modifications:

- Set Source \( s=(k,j,s_j) \), Destination \( d=(m,d_j,d_m) \)
- Procedure LLMR(s, d)
  - if \( k < m \) then \( d_1=\left\lceil d_1 / 2^{s_j} \right\rceil, d_2=\left\lfloor d_2 / 2^{s_j} \right\rfloor \)
  - else if \( m > k \) then \( s_1=\left\lfloor s_1 / 2^{j_k} \right\rfloor, s_2=\left\lceil s_2 / 2^{j_k} \right\rceil \)
  - if \( \text{dist}(s,d) \leq 3 \) return 0
  - \( k=\min(k,m) \)
  - if \( \text{dist}(s,d) > 3 \)
    - for \( i=0 \) to 6 do
      - \( P_i(s) = (k,j; \left\lceil s_1 / 2^i \right\rceil, \left\lfloor s_2 / 2^i \right\rfloor) \)
      - \( P_i(d) = (k;j; \left\lceil d_1 / 2^i \right\rceil, \left\lfloor d_2 / 2^i \right\rceil) \)
    - if \( i \neq \text{dist}(P(s), P(d)) \)
      - then \( \text{feasible}[i]=\infty \)
      - else \( \text{feasible}[i]=\| \)
  - \( \text{FEASIBLE}=\{ \text{feasible}[i]|\text{feasible}[i] \neq \infty \} \)
  - return \( \text{FEASIBLE} \)

**Definition 2:** Let distinct nodes \( s \) and \( d \) be in \( PM[n] \) which are in level \( j \) and \( k \) respectively. \( P^s_j(s) \) and \( d \) are in the same level where \( 0 \leq k \leq j \). The Procedure LLMR and Theorem 9 can find the least level \( \lambda \), \( \mu \in \text{FEASIBLE} \) in which the path between \( P^s_j(s) \) and \( d \) is the shortest path comparing with paths in the other levels. We have

\[
\text{rise}_\lambda(s,d) = \lambda, \quad \text{rise}_\mu(s,d) = \mu, \quad \text{when} \quad \lambda \leq \mu \\
\text{rise}_\lambda(s,d) = \mu, \quad \text{rise}_\mu(s,d) = \lambda, \quad \text{when} \quad \mu \leq \lambda
\]

4.2 The Node-Disjoint Routing Algorithm

**Source s(h,s_h,s_1)**
**Destination d(h,s_d,d_1)**
- if LLMR(s,d)=0 then
  - rise_level_1=1
  - rise_level_2=0
- else
  - rise_level_1=\( \max\{ j|j=\text{LLMR}(s,d) \} \}
  - rise_level_2=\( \max\{ j|j=\text{LLMR}(s,d) \} \}
- Parallel Routing
  - for \( h=1 \) to rise_level_{1}:
    - path 1 uses one uplink to the ancestor of \( s \)
    - if path 2 and path 3 do not reach rise_level_2, then path 2 goes to the nearest son of the neighbor of \( P^s(j) \) and uses an uplink to a neighboring node of \( P^s(j) \)
    - path 3 goes to the nearest son of the other neighbor of \( P^s(j) \) and uses an uplink to a neighboring node of \( P^s(j) \)
- path 1 uses X-Y routing or Y-X routing to \( P^s(d) \)
- path 2 uses X-Y routing or Y-X routing to a neighboring node of \( P^s_{\text{rise}_\lambda(d)} \)
  - path 3 uses Y-X routing or X-Y routing to the other neighboring node of \( P^s_{\text{rise}_\lambda(d)} \)
  - for \( \text{rise}_\lambda \) down to 1
    - path 1 uses one downlink and goes to the descendant node \( r \) of current node
    - if \( b<\text{rise}_\lambda \), then
      - path 2 uses one downlink and goes to a neighboring node of \( r \)
      - path 3 uses one downlink and goes to the other neighboring node of \( r \)
- end of the algorithm

4.3 Correctness

We first present the following theorem which states the maximum steps required by constructing three node-disjoint parallel paths in \( PM[n] \).

**Theorem 10:** For any pair of distinct nodes in \( PM[n] \), the lengths of path1, path2 and path3 are no more than \( 4n-2 \) by using the node-disjoint routing algorithm.

See Figure 10, because every node excepting apex in \( PM[n] \) has at least two siblings, we have the next corollary from Theorem 10 and the algorithm in section 4.2.

**Corollary 11:** By using our algorithm, we can construct three node-disjoint paths between any pair of distinct nodes in \( PM[n] \).

From the above, we actually have:

**Corollary 12:** An \( O(n) \) algorithm is given for constructing three node-disjoint paths between any two nodes in \( PM[n] \) and their lengths are no more than \( 4n-2 \).
5 Conclusions

In this paper, we proposed an efficient routing scheme (LLMR) in the pyramid networks. We developed two algorithms which can obtain end-to-end shortest path and fault-tolerant routing. The former can be modified for finding the node-disjoint paths in wormhole routing. Thus, it can tolerate up to three vertex and/or edge faults as well as take at most $4n-2$ steps between any pair of distinct nodes in $PM[n]$.

Compared with the results proposed by Feng Cao et al., where at most $(10/3)n+6$ steps are required between any pair of distinct nodes, we can obtain $4n - 2\leq(10/3)n+6$ for $n\leq12$. In fact, Table 1 shows that our scheme is better than that proposed by Feng Cao when $n<12$ and explicitly worse when $n\geq14$. Nevertheless, the number of nodes in the base of the a pyramid becomes extremely large (about more than $2^{30}$ nodes) when $n\geq14$. It is excessively difficult to implement such a gigantic number of PEs in a parallel computer or a VLSI at present. Even thought it is possible for such implementation, their time complexity will be much greater than ours. Moreover comparing with $O(n^2)$ time complexity obtained by Feng Cao et al., we realize that LLMR algorithm is very efficient due to its $O(n)$ time complexity.

Finally, we discuss the load balancing properties. The routing algorithm will release the apex from any transmission load for almost all pairs of nodes. Furthermore, it reduces enormously the traffics in higher levels of a pyramid.

6 References


<table>
<thead>
<tr>
<th>N</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>12</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes in base</td>
<td>16</td>
<td>64</td>
<td>256</td>
<td>4096</td>
<td>65536</td>
<td>262144</td>
<td>1048576</td>
<td>16777216</td>
<td>67108864</td>
<td>10737418234</td>
</tr>
<tr>
<td>Proposed</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>22</td>
<td>30</td>
<td>34</td>
<td>38</td>
<td>46</td>
<td>50</td>
<td>58</td>
</tr>
<tr>
<td>Feng Cao et al.</td>
<td>13</td>
<td>16</td>
<td>20</td>
<td>26</td>
<td>33</td>
<td>36</td>
<td>40</td>
<td>46</td>
<td>50</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 1 : Maximum steps for our algorithm and algorithm in [2].